

# LEADPOST

Francisco Palomares Barrios

Director: David Sánchez

Grado Multimedia

Junio 2014

## Resumen

LeadPost es una plataforma online de divulgación arquitectónica. La web tiene tres grandes apartados, Home, Productos y Social.

- **Home:** es donde se publican resúmenes de los proyectos más valorados de la red, del día anterior. Estos resúmenes se hacen a partir de entradas de blogs de arquitectura, a la cual siempre se puede acceder para más información.
- **Productos:** aquí de forma similar a los proyectos, las empresas relacionadas con el sector podrán publicar sus productos. Además se puede acceder al perfil de cada empresa, y contactar con ella para pedir presupuestos.
- **Social:** este es el apartado donde aparecen los proyectos subidos por usuarios arquitectos que sean autores de sus proyectos. También cabe la posibilidad de subir los proyectos como un despacho de arquitectura, al cual pueden estar afiliados usuarios. Los usuarios pueden seguir a otros, para ver sólo los proyectos de las personas o despachos que les interesen.

Los proyectos tanto del apartado Home como los de Social, cuentan con un filtrado que llamamos **canales**, que son las cinco grandes categorías que se encuentran en la parte superior de la página (Arquitectura, Sostenibilidad, Espacio urbano, Rehabilitación e Innovación). Éstos, también están clasificados en otros tres niveles de categorías, que llamaremos: **categorías**, **subcategorías** y **etiquetas**. Los proyectos van enlazados a una o más etiquetas, y éstas a su vez a una subcategoría, la cual está dentro de una categoría; y van enlazados a los tres niveles.

Cuando abrimos un resumen de proyecto o un producto, lo llamaremos **lead**; este término proviene del gremio periodístico y se utiliza para referirse a la “entradilla” de una noticia, esas dos o tres líneas que resumen su esencia y que cualquier lector lee para determinar si le interesa o no. Este lead viene provisto de una serie de partes:

- Imágenes: consta de seis imágenes cada lead.
- Información: que se divide en:
  - o El título del proyecto
  - o La persona que ha creado el lead.
  - o La fuente original, el blog.
  - o El enlace al post original.
  - o El nombre del arquitecto.
  - o What: explica de que trata.
  - o Where: explica el entorno donde se localiza.
  - o How: explica cómo se ha construido.

El criterio que se sigue para publicar en el apartado Home viene dado en función de un ranking de blogs de arquitectura propio de la plataforma; según se visualicen leads, se compartan en redes sociales, se acceda al post original, etc; éste obtiene una puntuación, que a su vez hace que el blog de origen obtenga la suya. De este modo se obtiene el **ranking de blogs** que más interesan a los usuarios; y las publicaciones solo se realizan de los 15 primeros blogs. El ranking no es un ranking estático, se va renovando cada dos semanas con nuevos blogs.

El aspecto general que tendría la plataforma totalmente acabada sería el que se refleja en la siguiente imagen

Home
Social
Products

English
sign up
login

Architecture
Sustainability
Urban space
Rehabilitation
Innovation

1 Archdaily  
2 Designboom  
3 I Like Architecture  
4 Detail  
5 HIC arquitectura  
6 Plataforma arquitectura  
7 Inhabitat  
8 Architzier  
9 Archello  
10 Pasajes arquitectura  
11 Plataforma arquitectura  
12 Inhabitat  
13 Architzier  
14 Archello  
15 Pasajes arquitectura

Register and be the first to receive updates

04may
01may
30apr
29apr
28apr
23apr
24apr

Envelope
Typology
Materials
Spaces
Interior
Structure
Furniture
Status

Join us!

Comments

About
Team
Privacy Policy

LeadPost  
architecture on the move

f t y+
Powered by KPB

En la parte superior izquierda encontramos el menú principal con los tres grandes apartados de la web (Home, Social, Products). En la parte superior central se sitúan los canales temáticos. En la zona superior derecha tenemos el acceso para los usuarios y el idioma de la web. En el centro de la web tenemos los proyectos publicados, a su lado izquierdo el ranking de blogs, y a su derecha las categorías desplegables con tres niveles.

# 1 Índice

Resumen .....	0
1. Introducción .....	4
1.1 Equipo.....	5
2 Objetivos .....	6
2.1 Objetivos generales.....	6
2.2 Objetivos específicos .....	6
2.3 Objetivos personales.....	6
3 LeadPost: Descripción general .....	7
3.1 Evolución de la idea .....	7
3.2 Público objetivo .....	7
3.3 Servicios .....	8
3.4 Funcionalidades .....	9
3.5 Calendario.....	10
4 Diseño .....	11
4.1 Mockups .....	11
4.2 Logotipo, color y tipografías.....	16
4.3 Diseño web definitivo .....	19
5 Desarrollo.....	31
5.1 Herramientas utilizadas .....	31
5.1.1 MySQL Workbench.....	31
5.1.2 Moqups.....	32
5.1.3 Sublime Text .....	33
5.1.4 Laravel .....	34
5.1.5 Bootstrap .....	35
5.1.6 Isotope.....	36
5.1.7 Otros .....	36
5.2 Base de datos .....	37
5.3 Programación en Laravel .....	43
5.3.1 Paquetes adicionales .....	43
5.3.2 Base de datos .....	44
5.3.3 Rutas.....	46
5.3.4 Filtros .....	46
5.3.5 Modelos .....	48
5.3.6 Controladores .....	50
5.3.7 Vistas .....	56

5.4	Programación en JavaScript .....	59
5.5	Diseño centrado en el Usuario .....	70
5.5.1	Evaluación heurística .....	70
6	Conclusiones .....	80
7	Bibliografía .....	81
7.1	Documentación de laravel.....	81
7.2	Documentación de Bootstrap .....	82
7.3	Documentación de Isotope: .....	82
7.4	Evaluación heurística .....	83
7.5	Otras consultas .....	83

## 1. Introducción

Durante el año 2012 estuve colaborando en la start-up (e)co UPC, equipo de investigación y diseño de prototipos de viviendas del futuro asociado con la Universidad Politécnica de Cataluña. Yo me encargaba de la web y la postproducción de vídeos. Por aquel entonces conocí a Aitor Iturralde, estudiante de arquitectura, el cual, un año después, terminaba su grado, y junto con dos compañeros más se les ocurrió la idea de crear una biblioteca de proyectos de arquitectura online. Acudieron a mí, por si me interesaba formar parte del proyecto. En esos momentos yo carecía de muchos de los conocimientos necesarios para hacer una plataforma web de esa envergadura, y lo sabía, pero aun así decidí meterme de lleno en el proyecto.

En mi desconocimiento de cómo realizar el proyecto de la mejor manera posible, comencé todo desde cero, teniendo una idea más o menos clara de lo que queríamos hacer, y así empecé a realizar la base de datos y programar la web en PHP.

A medida que se iban obteniendo resultados, e iba avanzando el proyecto, también lo hacía la idea, la cual ha ido evolucionando considerablemente desde el inicio, empezó siendo una biblioteca de proyectos, para terminar siendo una red social de arquitectura. Eran cambios estructurales demasiado grandes; tal y como se había desarrollado no servía el trabajo realizado, la base de datos era demasiado cerrada y el código PHP habría que reescribirlo. Tenía que buscar una forma de que esto no volviese a ocurrir, de que la plataforma pudiese crecer de forma que la base de datos y estructura no la limite. Así que en grupo estuvimos varios días replanteando la plataforma, pensando en todas las posibilidades que nos pudiésemos encontrar, para de ese modo yo poder plantear una buena base de datos. Ahora el problema estaba a la hora de desarrollar la plataforma, tenía que buscar una manera que me permitiese realizar el nuevo trabajo de forma más rápida y mejor, y no tener que programar todo desde cero, así que decidí aprender a utilizar el *framework* de PHP, Laravel, que parecía estar tomando fuerza en la comunidad PHP. Una vez tomado el nuevo rumbo, comencé a desarrollar de nuevo la plataforma, hasta el día de hoy.

La plataforma pretende ser un medio de comunicación de temática arquitectónica; en la cual se publican cada día quince post de proyectos de arquitectura los cuales son seleccionados de la siguiente manera: existe un ranking de los mejores blogs que se renueva cada dos semanas y de estos blogs se publica el post mejor valorado del día anterior. Los usuarios pueden interactuar con esas publicaciones, votándolas, guardándolas como favoritos, compartiéndolas, etc. Además habrá un apartado donde los usuarios puedan publicar sus propios proyectos; podrán seguir a otros usuarios, empresas, etc. La plataforma también ofrece un servicio para las empresas, donde podrán promocionar sus productos relacionados con el sector de la arquitectura.

Para el proyecto académico se presenta un prototipo funcional de la plataforma, en el cual se encuentra operativo las funciones más importantes. Se puede acceder a él a través del siguiente enlace:

<http://leadpost.eu/>

## 1.1 Equipo

El equipo de este proyecto está formado por:

- Jordi Gomar Manresa – Arquitecto.
- Sergi Valero Muñoz – Arquitecto.
- Aitor Iturralde Martín – Arquitecto.
- Francisco Palomares Barrios – Diseñador Multimedia.


Los tres arquitectos, se encargan de generar los contenidos, gestionar los colaboradores, así como la acción en las redes sociales; mientras que yo, Francisco Palomares Barrios, me encargo del diseño y programación de la plataforma.

El equipo se completa con colaboradores que suben contenido a la plataforma.




Más datos sobre las personas del equipo que forman LeadPost se pueden encontrar en el siguiente enlace:


<http://leadpost.eu/team>

### Team




Francisco Palomares Barrios


  




Jordi Gomar Manresa




Aitor Iturralde Martin

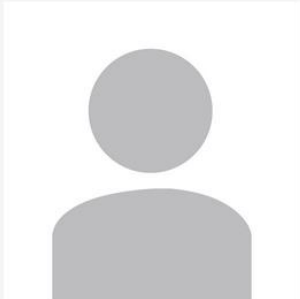


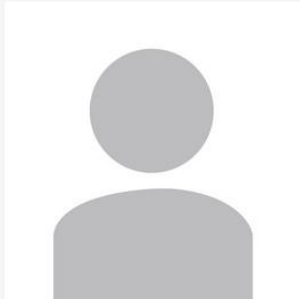



Sergi Valero Muñoz

### Collaborators









## 2 Objetivos

Al tratarse de un proyecto en continuo cambio y expansión, es muy posible que los objetivos vayan cambiado con el tiempo, e incluso surjan nuevos objetivos en los cuales no se habían pensado.

### 2.1 Objetivos generales

Realizar un prototipo de la plataforma, en el cuál se puedan visualizar los leads de forma cómoda, tanto desde un ordenador de escritorio como desde cualquier dispositivo móvil; además exista un primer filtrado de leads.

### 2.2 Objetivos específicos

El núcleo central de la plataforma, radica en su **base de datos**; por lo que crear una escalable y sólida base de datos era fundamentalmente necesario; y de este modo poder adaptarse a un crecimiento continuo en sus funcionalidades.

Hoy en día la cantidad de dispositivos que existen en el mercado donde poder visualizar contenido web es inmensa, por este motivo es importante realizar una buena adaptación de la plataforma al máximo de **dispositivos móviles** y resoluciones de pantalla.

### 2.3 Objetivos personales

Uno de mis objetivos principales para poder desarrollar esta plataforma de una forma más eficiente era dominar el **framework de PHP Laravel**. Desde la versión 4 se está convirtiendo en una buena opción a la hora de programar en PHP, ya que recopila las mejores funcionalidades de otros *frameworks* como Symfony, Ruby on Rails, Sinatra, etc.



### 3 LeadPost: Descripción general

El contenido arquitectónico se encuentra disperso por la red, de forma que la búsqueda de información muchas veces son lentas y poco efectivas. LeadPost ofrece a los usuarios una biblioteca online de la actualidad arquitectónica, agrupándola, resumiéndola y etiquetándola para sus diferentes filtrados de búsquedas.

El objetivo de LeadPost es ofrecer el **contenido arquitectónico más valorado** de la red de la forma más ágil, rápida y eficaz posible; convirtiéndose en una herramienta básica tanto para profesionales como para aficionados del sector.

LeadPost ofrecerá a las empresas del sector arquitectónico una **publicidad segmentada**, de forma, que estas elijan al público al cual va destinada su publicidad. Para conseguir esto, recogemos datos de los usuarios que nos hacen saber sus preferencias e intereses, y se les mostrará el producto publicitario de las empresas que hayan seleccionado estos mismos intereses.

#### 3.1 Evolución de la idea

La idea inicial de LeadPost, era crear una plataforma web con el objetivo de almacenar proyectos arquitectónicos extraídos de los blogs de arquitectura más visitados, para que se convirtiese en una especie de **biblioteca de proyectos** a la cual se puede acceder para ver los últimos proyectos o para extraer ideas de todos los trabajos que hay, con un sistema eficaz de búsqueda.

Pero esta idea con el paso de los meses fue evolucionando; ya no solo sería una biblioteca de proyectos, también tendríamos un apartado de **productos arquitectónicos** publicados por empresas interesadas en el sector.

Y por último, la idea evolucionó hasta el punto de convertirse en **una red social**; para que hubiese usuarios que pudiesen subir sus propios proyectos, que pudiesen seguir a despachos de arquitectura que le interesen, o a empresas del sector, o a blogs de arquitectura; y además la plataforma contaría con un ranking de los mejores blogs de arquitectura, según el contenido que vayan publicando.

#### 3.2 Público objetivo

LeadPost es una plataforma dedicada al mundo de la arquitectura y la construcción. No obstante, el público objetivo se puede clasificar dentro de varios niveles, y por lo tanto nuestras estrategias de marketing tendrán que cambiar según el usuario.

- **Estudiantes de arquitectura:** utilizarán la plataforma con frecuencia para informarse de la actualidad arquitectónica del momento y para construirse sus propias bibliotecas de proyectos de donde tomar ideas, éstos no pagarían y si lo hacen serían pocos de ellos.
- **Arquitectos:** será el público principal. Utilizaran la plataforma como herramienta profesional para promocionar sus proyectos y para buscar la actualidad en el sector.
- **Otros profesionales:** agentes dentro del sector de la construcción (ingenieros, diseñadores, constructores, etc) que utilizarán la plataforma como buscador de actualidad y para informarse de productos industriales.
- **Aficionados:** destinaran pocos recursos y tiempo.
- **Organismos:** como universidades de arquitectura, les interesará crear consciencia de marca y posicionarse dentro de la comunidad arquitectónica.

En base a nuestro público objetivo han surgido diferentes tipos de usuarios en la plataforma, con diferentes funcionalidades:

- **Usuario invitado:** es aquel que no está registrado en la plataforma.
- **Usuario fremium:** es el usuario que tiene el registro básico.
- **Usuario premium:** este usuario hace un pago trimestral por funciones extras en la plataforma.
- **Blog:** es gestionado desde uno o más usuarios, y puede subir leads sobre publicaciones.
- **Despacho de arquitectura:** es gestionado desde uno o más usuarios y puede subir sus proyectos.
- **Empresa:** es gestionado desde uno o más usuarios y puede subir sus productos.

### 3.3 Servicios

La plataforma ofrece diferentes servicios según el tipo de usuario:

- **Servicio de búsqueda en la biblioteca:** todo el contenido almacenado y categorizado según el material, tipología, elementos constructivos, etc; y accesible mediante un filtrado con tres niveles de categorías (categorías, subcategorías y etiquetas).
- **Servicio de visualización de leads:** LeadPost realiza resúmenes diarios de la información mejor valorada que se publica en la red, enfocados a los elementos característicos que definen cada proyecto y lo organiza en canales temáticos.
- **Servicio para usuarios individuales:** podrán hacerse sus propias bibliotecas con el contenido publicado, así como votar los blogs y los resúmenes realizados, para generar un ranking de los mejores blogs de arquitectura.
- **Servicio para profesionales:** permite utilizar la plataforma para divulgar sus proyectos, realizando resúmenes y etiquetándolos con el sistema de categorías de LeadPost. Los usuarios se pueden registrar como miembros de despachos o empresas, donde se publiquen los trabajos de ésta.
- **Servicio para empresas de la construcción:** las empresas podrán anunciarse sólo para el público que les interesa de forma segmentada, apareciendo cuando los criterios de búsqueda del usuario coinciden con los determinados por la empresa. Podrán subir sus ofertas dentro de la sección de productos de la plataforma.

### 3.4 Funcionalidades

En este punto se va hacer un recorrido por todas las funcionalidades principales que ofrece la plataforma según los tipos de usuario.

	Invitado	Fremium	Premium	Blog	Despacho	Empresa
Registro/login*	✓	✓	✓	✓	✓	✓
Visualización de leads /compartir en redes sociales*	✓	✓	✓	✓	✓	✓
Búsquedas por canales*	✓	✓	✓	✓	✓	✓
Búsquedas por nombre de proyecto*	✓	✓	✓	✓	✓	✓
Búsquedas por categorías, subcategorías	✓	✓	✓	✓	✓	✓
Visualización de perfiles de usuarios/empresas/blogs/despachos de arquitectura*	✓	✓	✓	✓	✓	✓
Visualización de proyectos de usuarios/despachos de arquitectura	✓	✓	✓	✓	✓	✓
Visualización de productos de empresas	✓	✓	✓	✓	✓	✓
Votar leads.	✗	✓	✓	✗	✗	✗
Realizar búsquedas hasta el nivel de etiquetas.	✗	✓	✓	✓	✓	✓
Seguir a otros usuarios, blogs, empresas o despachos de arquitectura.	✗	✓	✓	✓	✓	✓
Hacerse una biblioteca personal donde almacenar sus proyectos favoritos y categorizarlos como desee.	✗	✓	✓	✗	✗	✗
Mandar mensajes entre usuarios/blogs/empresas/despachos de arquitectura	✗	✓	✓	✓	✓	✓
Enviar invitaciones a usuarios para conseguir una temporada de premium	✗	✓	✗	✗	✗	✗
Crear una página de empresas/blog/despacho de arquitectura y administrarla	✗	✓	✓	✗	✗	✗
Subir proyectos como arquitecto*	✗	✗	✓	✗	✓	✗
Subir proyectos ajenos vinculados a un post de su web (en el caso que estén entre los 15 primeros del ranking sus publicaciones saldrán en el apartado Home).	✗	✗	✗	✓	✗	✗
Ver estadísticas, visitas de leads, estado en el ranking, etc.	✗	✗	✗	✓	✓	✓
Añadir/eliminar usuarios para gestionar la entidad	✗	✗	✗	✓	✓	✓
Crear tipos de usuarios administrativos de la entidad con diferentes permisos	✗	✗	✗	✓	✓	✓
Comprar paquete de funcionalidades	✗	✗	✗	✗	✓	✓
Subir productos.	✗	✗	✗	✗	✗	✓

\*Funcionalidades implementadas hasta el momento.

### 3.5 Calendario

Durante todo el proceso del proyecto, ha habido muchos retrasos. El más destacado es el momento en que se tuvo que comenzar de nuevo el planteamiento de todo el proyecto. Es por este motivo que las tareas en el archivo Project se extienden tanto en el tiempo. El proyecto se comenzó en el mes de marzo de 2013, pero se tuvo que replantear completo en el mes de octubre de ese mismo año desechando los avances programados hasta el momento.

Una de las metas conseguidas fue desarrollar un primer prototipo de la plataforma funcional antes del día 21 de mayo de 2013, que comenzaba el Construmat de Barcelona, feria especializada en materiales, equipamientos y servicios de construcción. Allí se iba a presentar el proyecto para obtener *feedback* de las empresas del sector. Pese a ello, este prototipo fue descartado por la gran falta de capacidad para resolver las nuevas necesidades que surgieron. Este prototipo aún se puede ver en el siguiente enlace: <http://leadpost.eu/antigua/>.

En los archivos anexos se adjunta un archivo de Project.

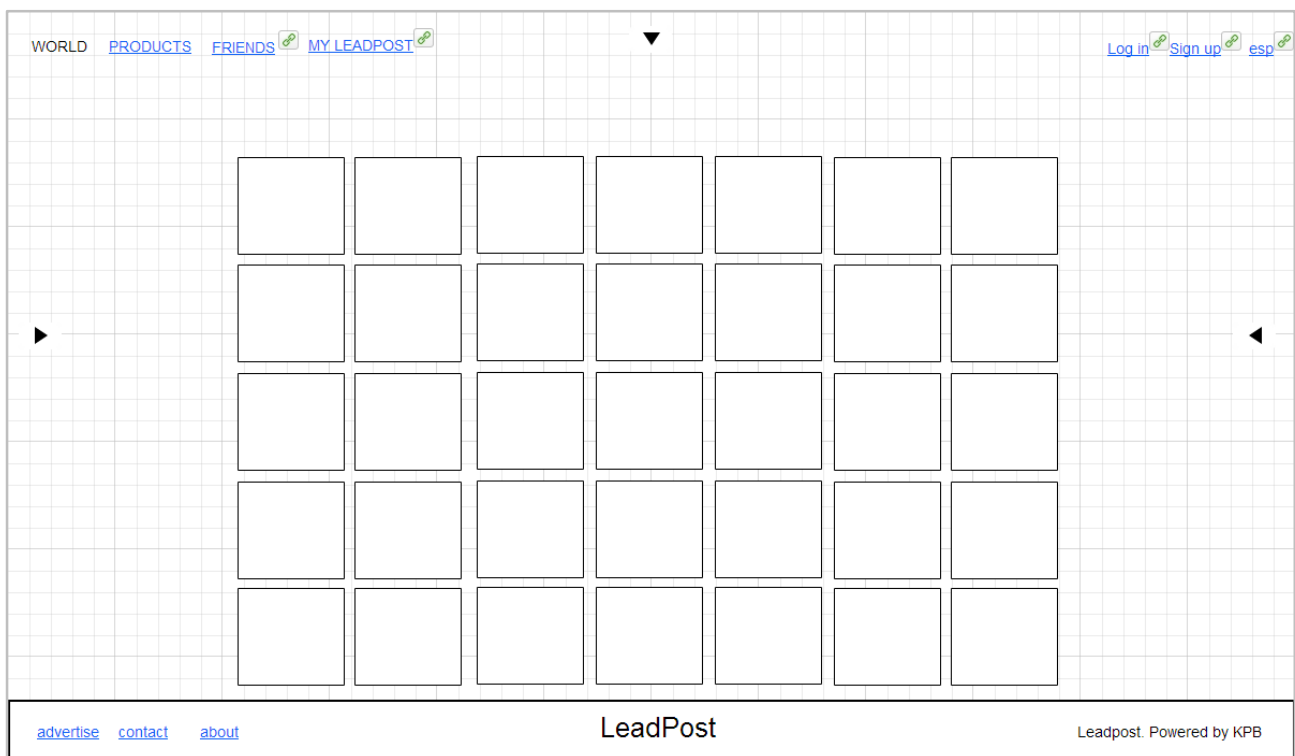
## 4 Diseño

En esta sección se verá primero los *mockups* realizados. Después se pasa al diseño corporativo de LeadPost, para por último ver el diseño final de la plataforma.

### 4.1 Mockups

Antes de comenzar a realizar el diseño definitivo sobre HTML y CSS, se han desarrollado *mockups* de las diferentes páginas de la plataforma. Muchas de estas pantallas han evolucionado o han sufrido cambios posteriormente una vez implementadas. A continuación se muestran los diseños más importantes:

- **Página de Home:** en la parte central se muestran los últimos proyectos publicados en forma de imágenes, con un *scroll* infinito que muestra los proyectos en orden cronológico. En la parte superior izquierda se encuentra el menú con los grandes apartados de la web; este menú está presente en todas las páginas; al igual que el menú superior derecho que se trata de los métodos para iniciar sesión, o cambiar el idioma; cuando un usuario ya ha iniciado sesión estos enlaces se sustituirán por otras opciones para él. De forma fija y permanente el pie de página siempre se mostrará, junto con él, el logotipo en su centro, claramente visible. A su lateral izquierdo se encuentra un menú de menos relevancia con páginas para la política de privacidad o contacto.



- **Página de inicio de sesión:** es un *popup* encima de todas las páginas de la plataforma, de forma que sea accesible cómodamente.

WORLD PRODUCTS FRIENDS MY LEADPOST

Log in Sign up esp

Mail

Password

Log in

advertise contact about

LeadPost

Leadpost. Powered by KPB

- **Perfil de usuario:** en el perfil de usuario se encuentra la imagen del usuario, y bajo este todos los botones de acciones sobre él. A su derecha los datos del usuario. En la sección 4.3 que se muestra el diseño definitivo se aprecian grandes cambios, ya que surgieron más necesidades para esta página.

WORLD PRODUCTS FRIENDS MY LEADPOST

usuario

My projects

Followers

Following

Messages

t

f

...

Name

Surname

Second surname

Country, city

Edad

Sexo

Entidad

Perfil profesional

Perfil profesional 2

web

mail

teléfono

descripción.....

advertise contact about

LeadPost

Leadpost. Powered by KPB

- **Subir un proyecto:**

- **1:** para subir un nuevo Lead, se pensó en realizarlo en cuatro pasos, para que al usuario no le supusiese un trabajo pesado de realizar. En este primer paso se indican los datos básicos del proyecto. En la parte inferior existen cuatro pequeños puntos que indican los diferentes pasos, y pulsándolos o haciendo scroll se puede cambiar de uno a otro.

WORLD PRODUCTS FRIENDS MY LEADPOST

Aitor Iturralde 7

ENG ESP +

titulo

arquitecto

web arquitecto

emplazamiento

LINK al post

LeadPost

advertise contact about

Leadpost. Powered by KPB

- **2:** el siguiente paso sería el de subir las seis imágenes que componen el lead, además del fotógrafo de cada imagen. Teniendo seis huecos donde poder arrastrar las imágenes o clicando sobre ellos para buscarlos con el explorador. Hay un input para el fotógrafo debajo cada imagen, y uno más grande debajo de todas las imágenes por si se tratase de un solo fotógrafo para todas.

WORLD PRODUCTS FRIENDS MY LEADPOST

Aitor Iturralde 7

photographer photographer photographer photographer photographer photographer

photographer

LeadPost

advertise contact about

Leadpost. Powered by KPB

- **3:** la tercera parte de subir un lead trata de rellenar en los diferentes idiomas los tres textos claves que componen el lead: *what*, *where* y *how*. De este modo se opta por tres grandes espacios para rellenar los textos; teniendo la posibilidad de cambiar el idioma a rellenar con un pequeño menú sobre ellos.

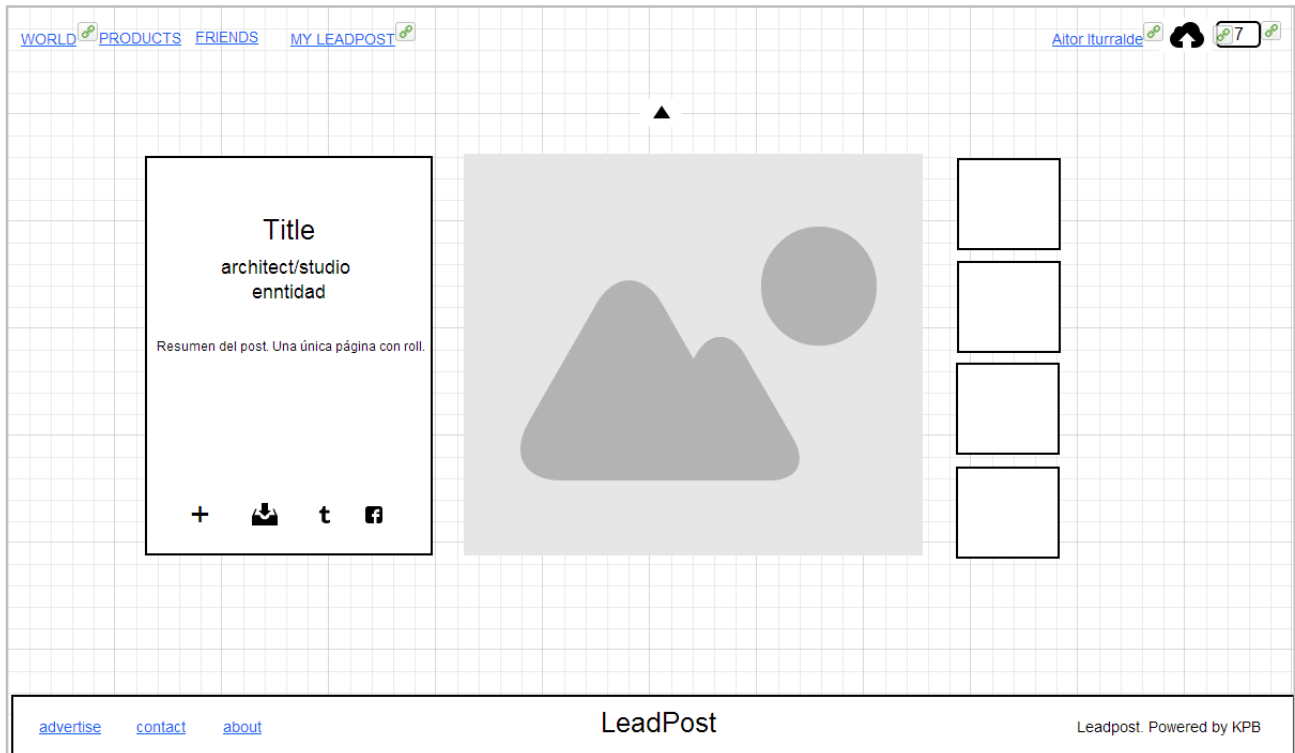
The screenshot shows the LeadPost interface. At the top, there are navigation links: [WORLD](#), [PRODUCTS](#), [FRIENDS](#), and [MY LEADPOST](#). On the right, the user's name 'Aitor Iturraide' is displayed next to a profile icon and a notification badge with the number '7'. Below the navigation bar, there is a language selection menu with buttons for 'ENG', 'ESP', and a '+' icon. The main area consists of three large rectangular boxes labeled 'WHAT', 'WHERE', and 'WHAT' from left to right. Above the boxes is an upward-pointing triangle, and below them is a downward-pointing triangle. At the bottom of the main area, there are three small dots, with the middle one being blue. The footer contains links for [advertise](#), [contact](#), and [about](#), the 'LeadPost' logo, and the text 'Leadpost. Powered by KPB'.

- **4:** por último es seleccionar las etiquetas que va a tener el lead, como es una categorización de tres niveles, creamos tres columnas, donde la primera se corresponda a las categorías, la segunda a las subcategorías una vez se seleccione categoría, y la tercera a las etiquetas que hay que marcar una vez se elija la subcategoría. Para llevar la cuenta de las etiquetas que han sido seleccionadas irían apareciendo debajo. Bajo esto el botón de subir el proyecto.

The screenshot shows the LeadPost interface with the tag selection area. The navigation bar and user profile are the same as in the previous screenshot. The language selection menu is still present. The main area now shows three columns of tags. The first column is labeled 'tupology', the second 'residential', and the third 'single house'. Below the third column, there is a list of selected tags: 'Single house', 'wood', and 'foster', each followed by a small 'x' icon. At the bottom of the main area, there is an 'UPLOAD' button. Below the button are three small dots, with the rightmost one being blue. The footer is the same as in the previous screenshot.



- **Visualización de lead:** cuando se abre un lead, en su izquierda se ve la información, de forma paginada pudiendo cambiar de información haciendo scroll con el ratón, o con los indicadores. Además de estar los enlaces para compartir en las redes sociales, votar, más información, etc. En la parte central y ocupando el mayor espacio las imágenes, que se pueden ir cambiando haciendo scroll con el ratón y con los marcadores que indican que imagen se está prevvisualizando. En la columna que queda en el margen derecho va ubicada la publicidad, los logotipos de empresas relacionadas con el proyecto.



## 4.2 Logotipo, color y tipografías

El logomarca de LeadPost, se compone de un grafismo de cuadrados junto al nombre de la plataforma, seguido del eslogan. Para el grafismo se intentó buscar algo que representase al diseño de la plataforma, siguiendo una estética minimalista y sencilla, éste se compone de cuatro cuadrados que representan las imágenes de los lead en la página principal de una forma simplificada.

Normal:



Monocromático:

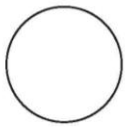


Negativo:



El color principal de la plataforma que se escoge es el azul, que junto al blanco da una impresión de limpieza y sencillez. Este color al ser el principal de la plataforma va ligado al canal más importante, Arquitectura, mientras que el resto de canales tienen vinculado otros colores:

- El color verde va vinculado al canal de sostenibilidad, por las connotaciones ecologistas que tiene este color.
- El amarillo va vinculado a Espacio urbano; es un color que tiene connotaciones de energía, de organización, que podrían ir bien asociadas a lo que los espacios urbanos se refieren.
- El rosa se vincula con Rehabilitación por sus connotaciones de calma, por representar a uno de los colores tradicionales de dormitorios antiguos; la mayoría de proyecto de rehabilitación son sobre edificaciones antiguas.
- El naranja va vinculado al canal de Innovación, ya que es un color que representa creatividad, y va vinculado a gente joven.



L: 100 R: 255  
a: 0 G: 255  
b: 0 B: 255

Hex: FFFFFFFF



L: 77 R: 188  
a: -1 G: 190  
b: -2 B: 192

Hex: BCBEC0



L: 37 R: 88  
a: 0 G: 88  
b: 0 B: 88

Hex: 585858



L: 0 R: 0  
a: 0 G: 0  
b: 0 B: 0

Hex: 000000



L: 64 R: 0  
a: -39 G: 174  
b: -49 B: 239

Hex: 00AEEF



L: 67 R: 72  
a: -80 G: 192  
b: 42 B: 88

Hex: 48C058



L: 37 R: 77  
a: 0 G: 1  
b: 0 B: 85

Hex: D3BA07



L: 46 R: 179  
a: 69 G: 24  
b: 24 B: 74

Hex: B3184A



L: 57 R: 219  
a: 71 G: 63  
b: 71 B: 3

Hex: DB3F03

La tipografía a utilizar es la **Helvetica Neue**, ya que es de palo seco, sin serifa. Las tipografías de palo seco ofrecen una lectura mejor en pantalla que las que tienen serifa, debido a la pixelación; por este motivo al ser una plataforma que se va a visualizar desde pantallas la Helvetica Neue es una buena opción. Se utiliza su versión *Thin* para los grandes títulos, de este modo queda una tipografía muy fina y limpia, y su versión *Roman* para los bloques de texto.

Helvetica Neue LT Com Roman:

ABCDEFGHIJKLMNOPQRSTUVWXYZ-  
VWXYZ  
abcdefghijklmnopqrstuvwxyz

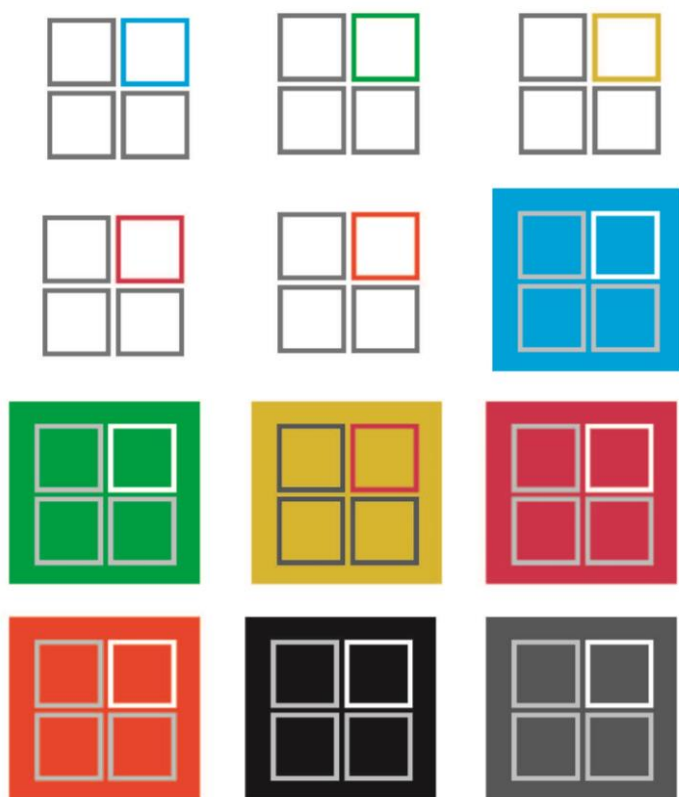
Helvetica Neue LT Com 35 Thin:

ABCDEFGHIJKLMNOPQRSTUVWXYZ  
abcdefghijklmnopqrstuvwxyz  
0123456789 +-( )@#

Ejemplo de tipografía dentro de cuadros de colores:



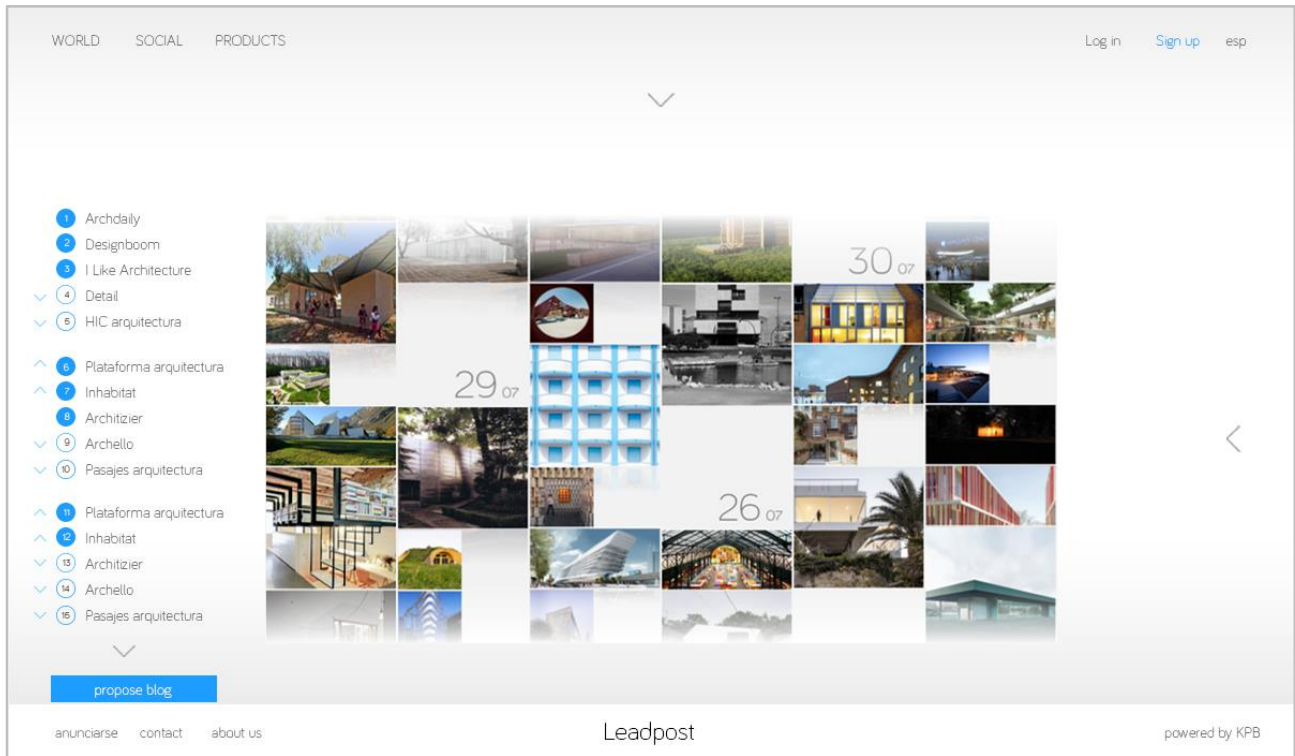
Grafismo del logotipo en diferentes colores y fondos:



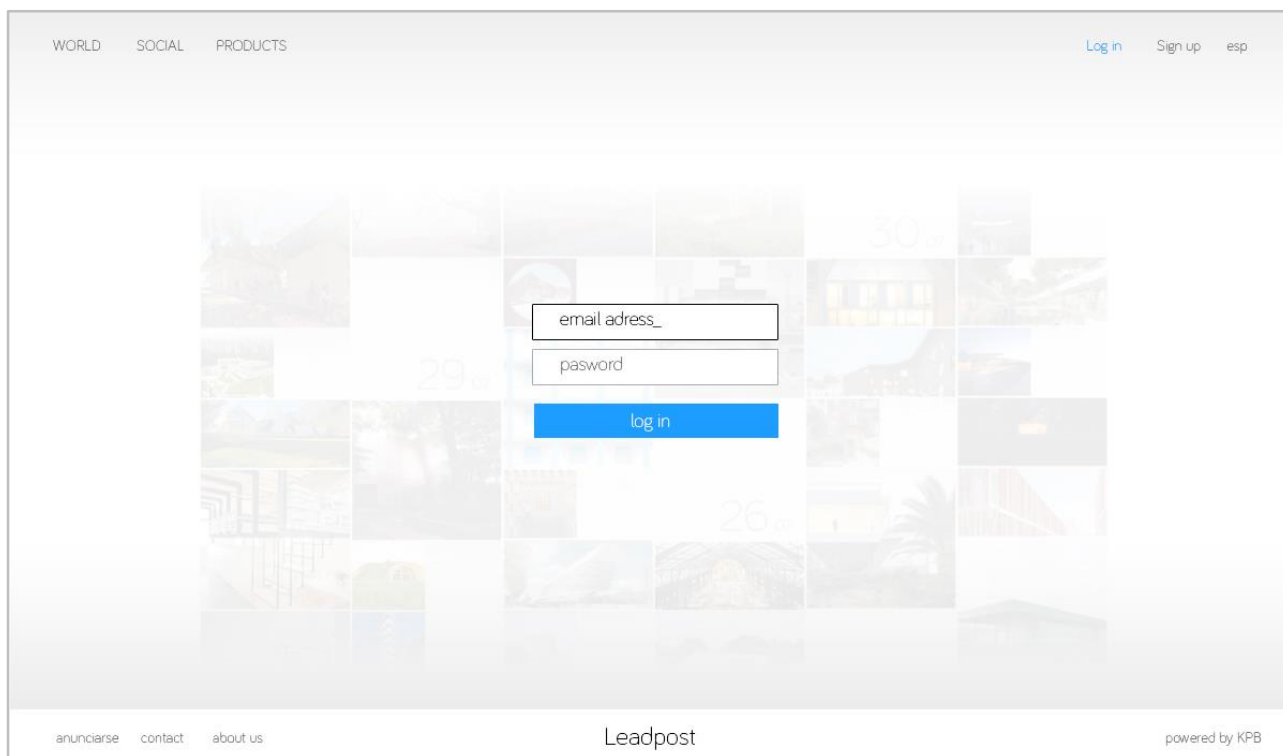
### 4.3 Diseño web definitivo

Una vez se tenían los *mockups* y el estilo de diseño a seguir, se procedió a realizar el diseño final de la web sobre Photoshop. Aunque este diseño ha cambiado considerablemente de lo que ahora nos encontramos en la web, ya que ha habido detalles que se han ido mejorando.

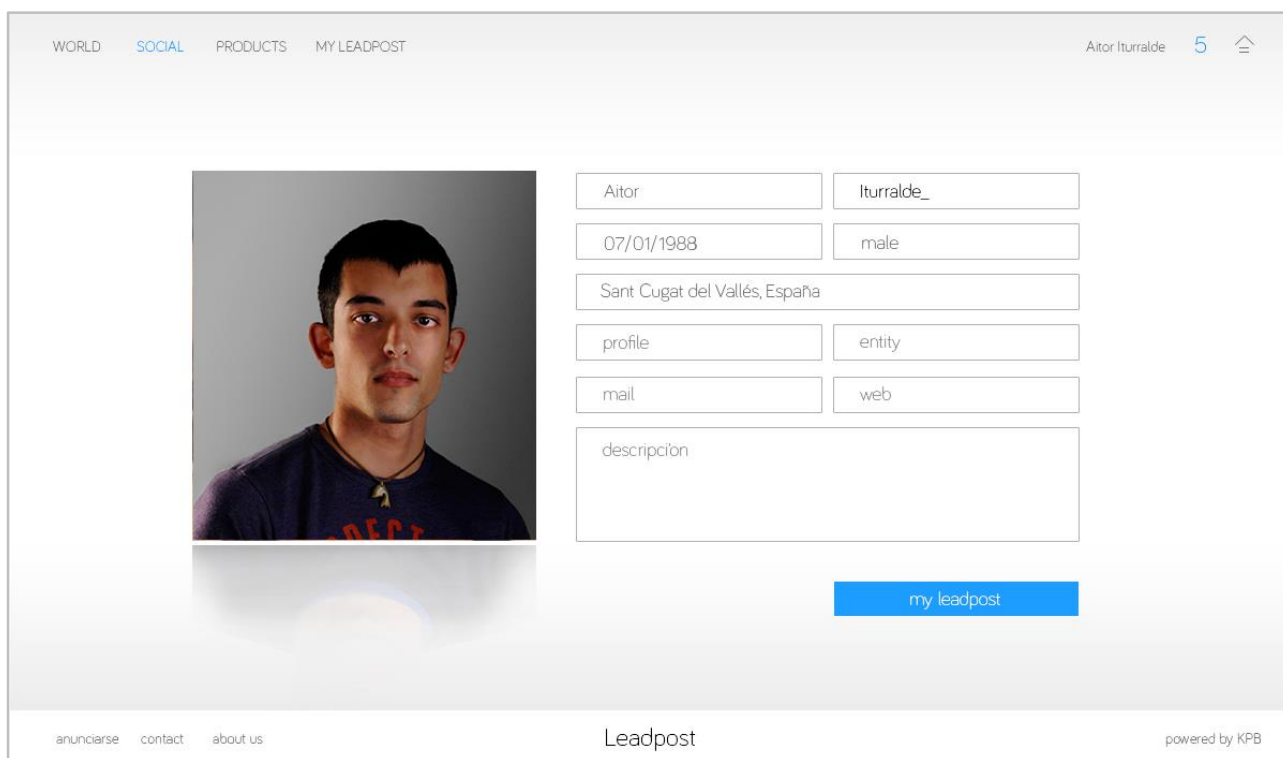
- Página de Home: su estructura siempre ha sido la misma; como se observa en el diseño los menús iban a ser ocultos con flechas que permitían desplegarlos. En el menú de los canales se ha descartado esta idea por tema de comodidad a la hora de usarlo; así que siempre esta visible.



- Página de inicio de sesión:

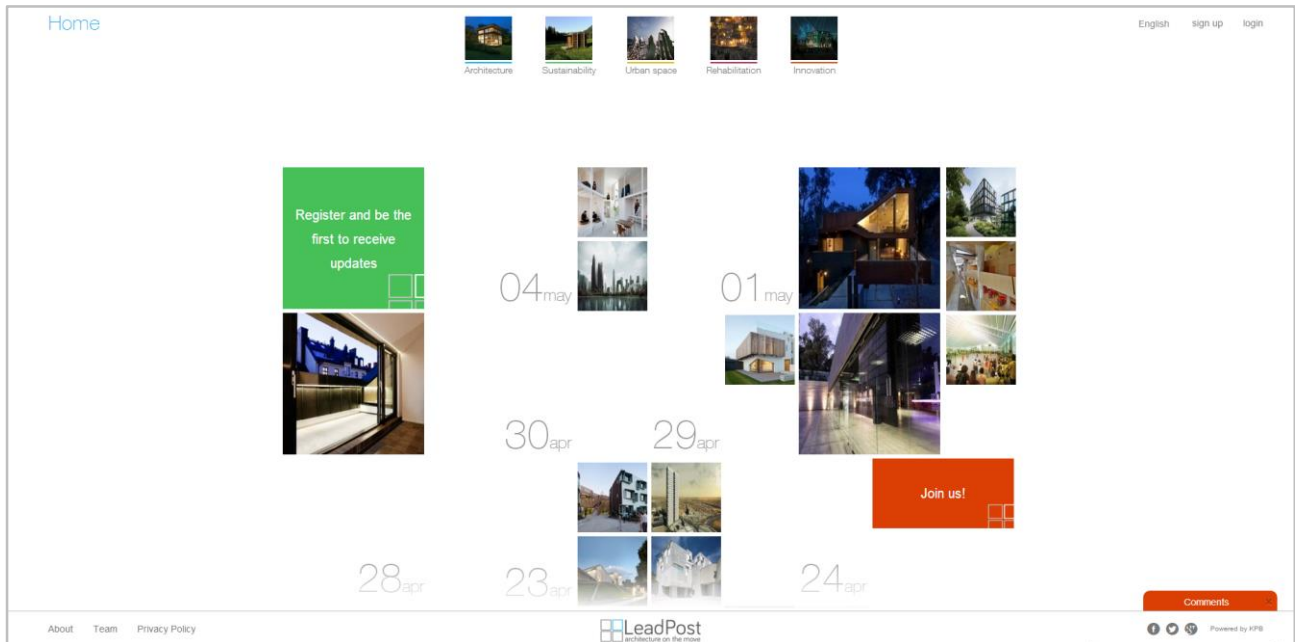


- Perfil de usuario: esta página es la que más cambios ha sufrido, ha pasado por muchas versiones una vez estaba la plataforma funcionando; ahora es completamente diferente a la idea inicial; debido a necesidades que han ido surgiendo: la necesidad de visualizar los leads del usuario, combinar el perfil con la edición de datos del propio usuario, etc.

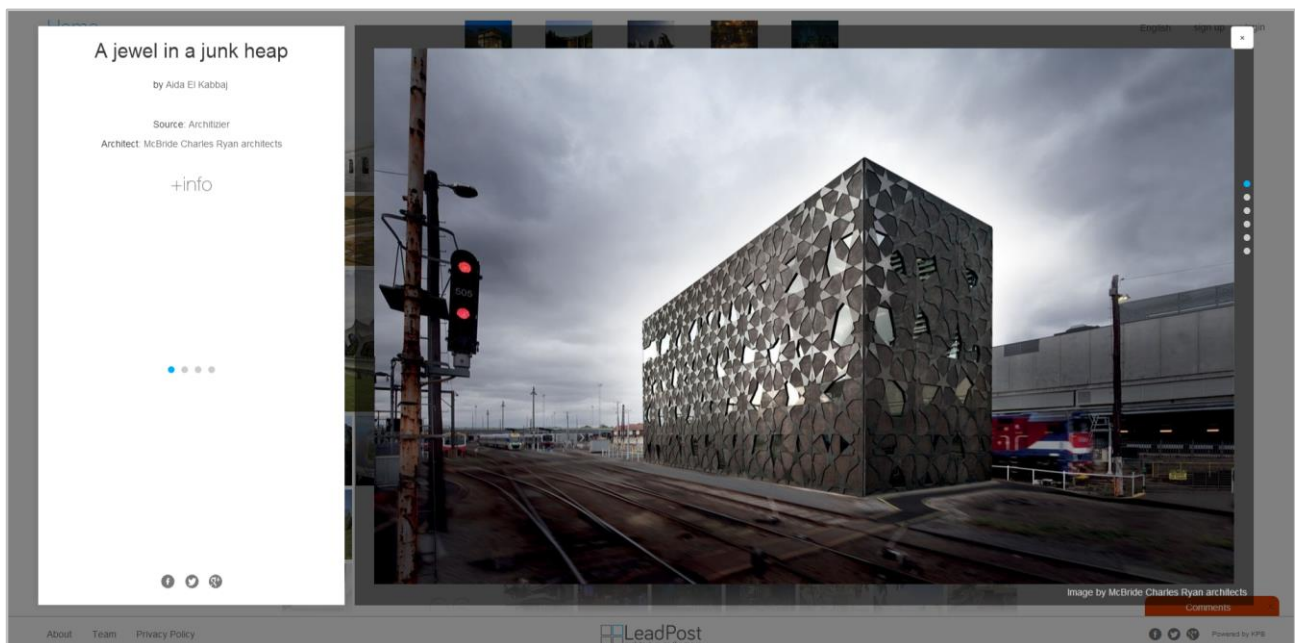


Se muestra el diseño definitivo que tiene la web programada y funcional:

- La página Home:

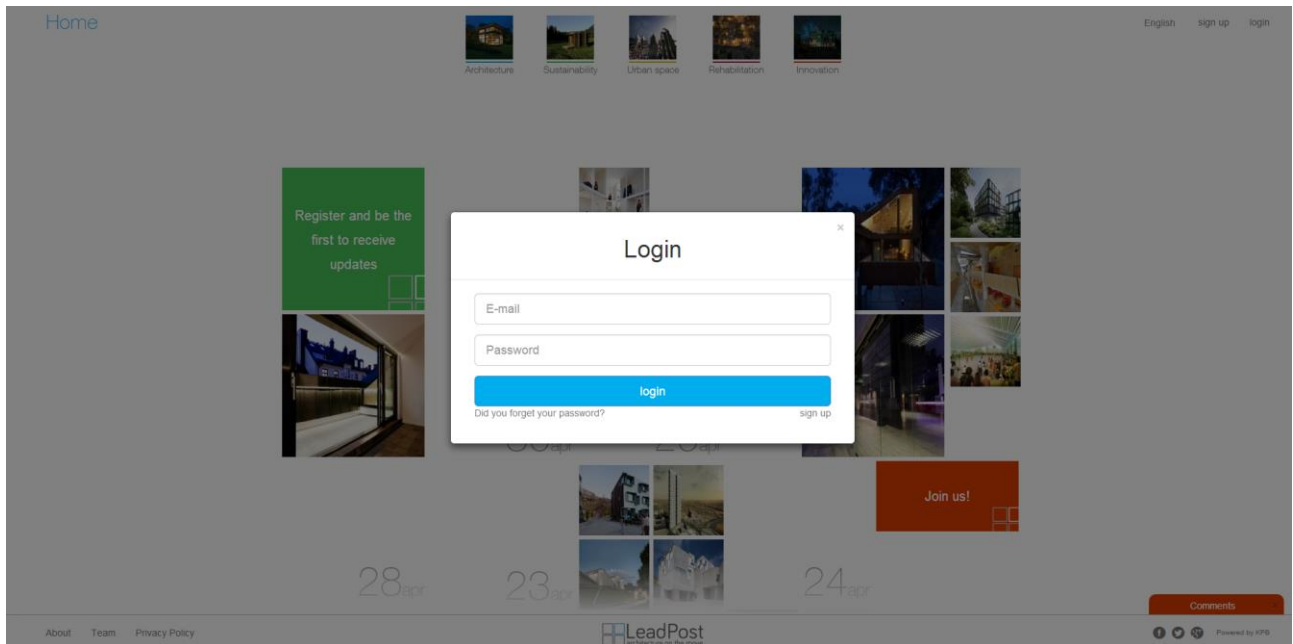


- Un lead abierto: este diseño ha pasado por muchas versiones, en un primer lugar las imágenes se abrían a un tamaño más reducido, ahora se aprovecha todo el espacio de pantalla para que la imagen cobré mayor importancia y su visualización sea más cómoda.

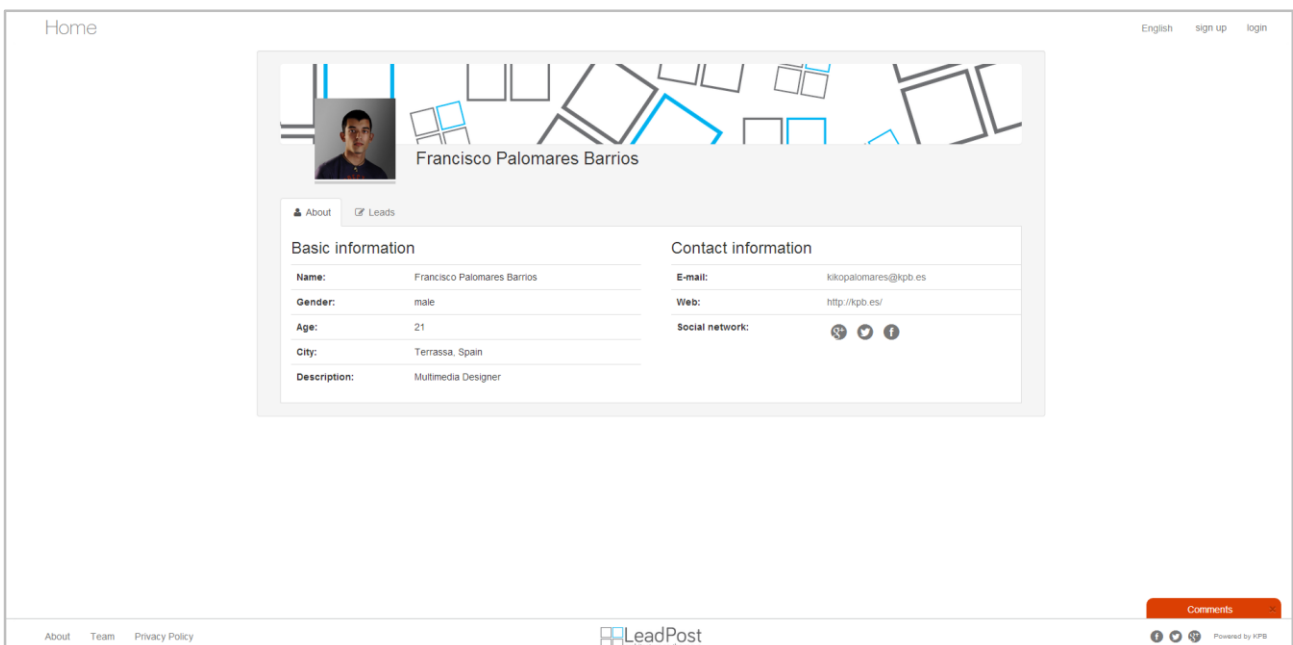




- Página de inicio de sesión:



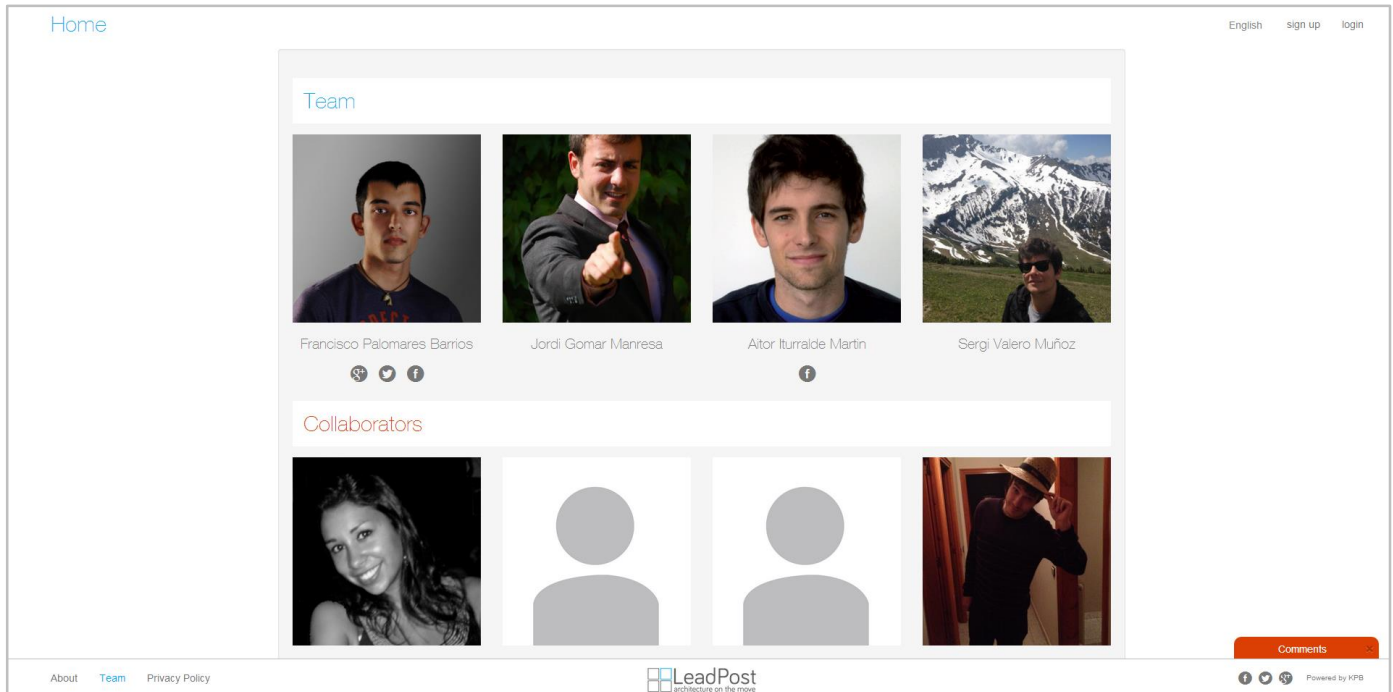
- El perfil de usuario: se ha optado por un sistema de *tabs* que permite tener toda la información del usuario en una sola página, de forma que se pueden ir añadiendo nuevas pestañas; cuando el usuario ha iniciado sesión y entra en su propio perfil le aparece una pestaña nueva donde puede editar sus datos de cuenta.





Una vez estando la web en línea, surgieron algunas páginas que no se contaba con ellas en un inicio, como por ejemplo las siguientes:

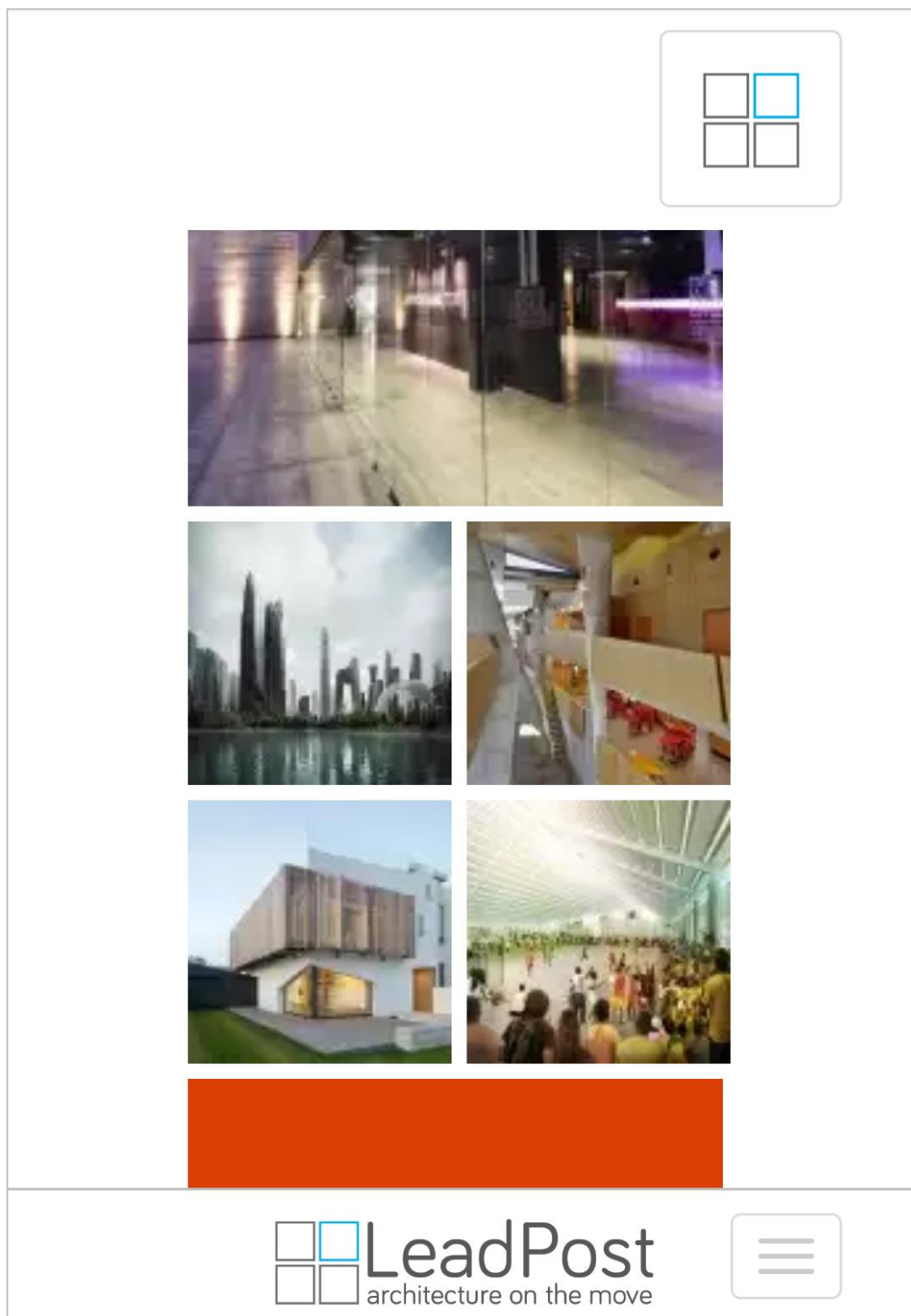
- Equipo:



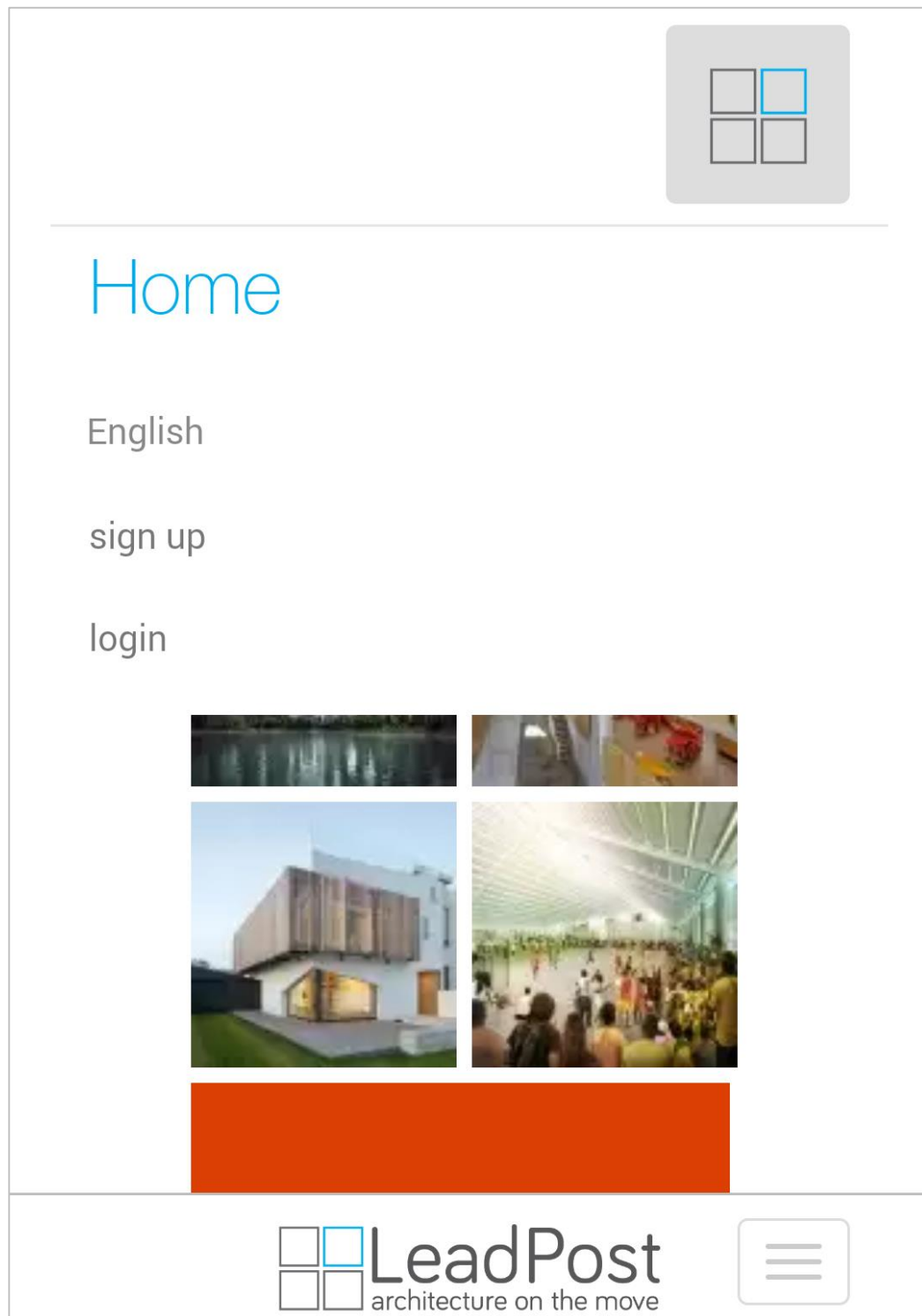
- Acerca de:



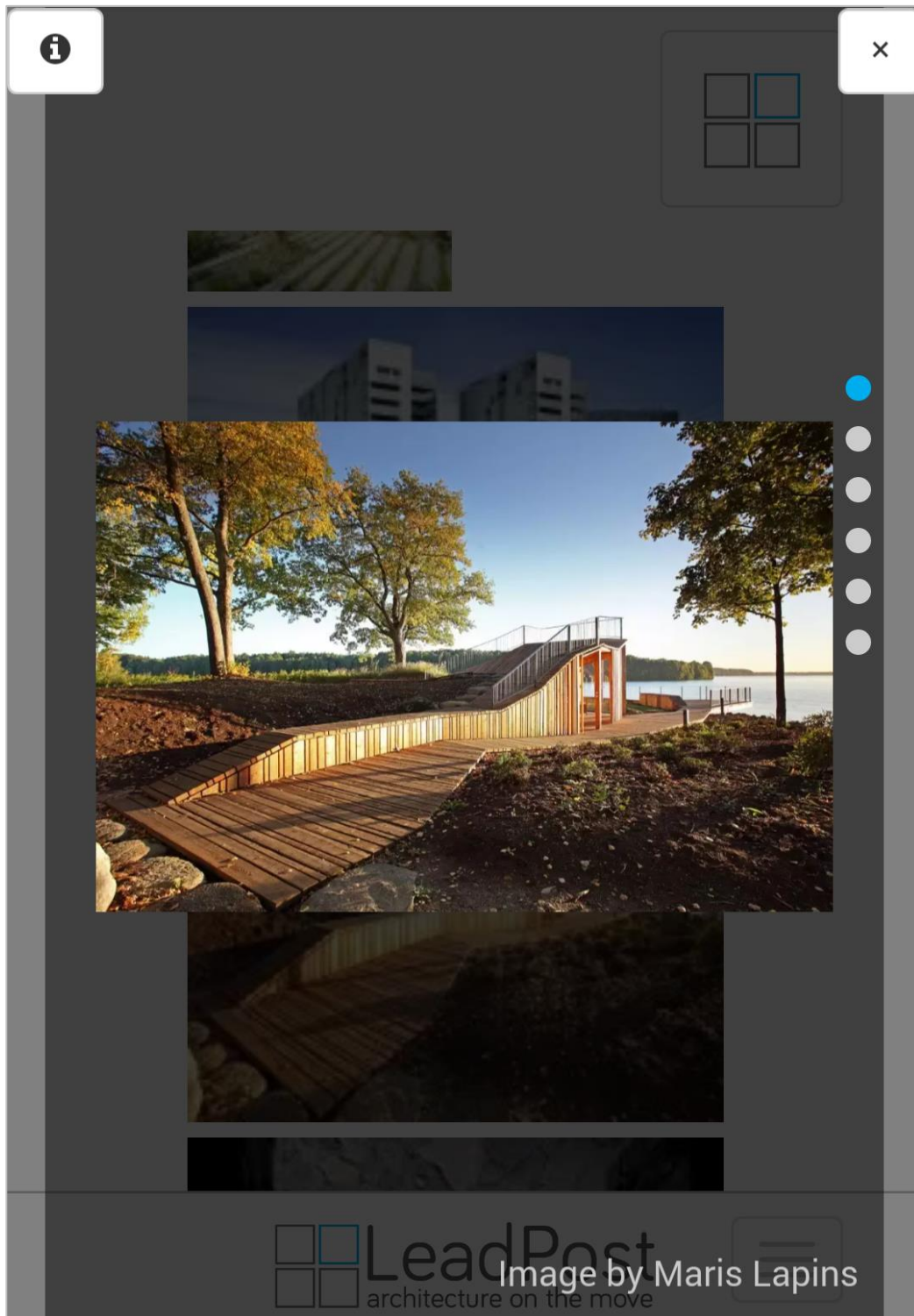
- La página de Home adaptada para móvil muestra los leads que quepan de forma centrada usando todo el ancho de pantalla; el menú queda colapsado en el botón superior con el grafismo del logotipo; cuando se pulsa se despliega hacia abajo.
  - o Menú plegado:



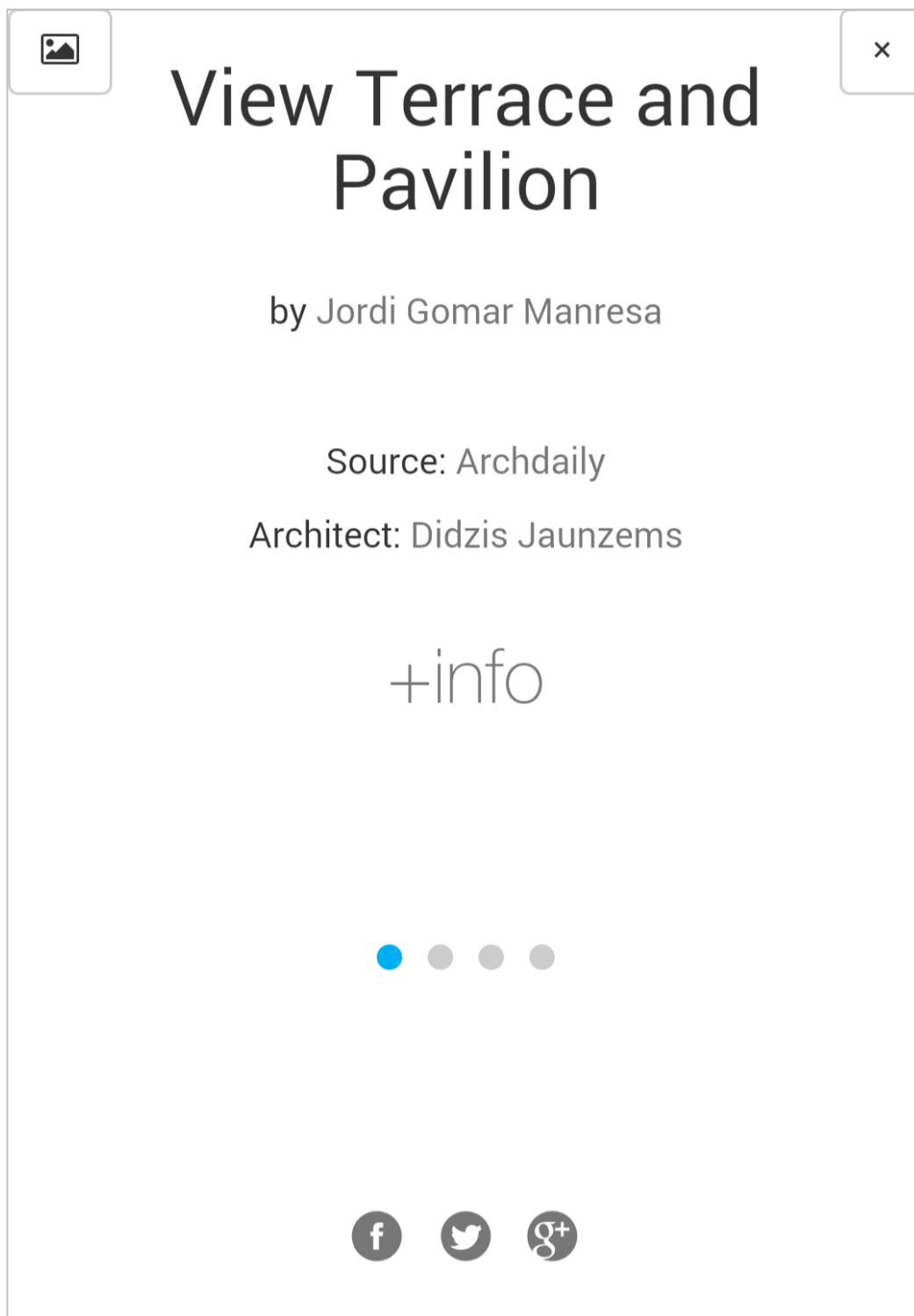
- Menú desplegado:



- Cuando se pulsa sobre una imagen para abrir un lead, lo primero que aparecen son las imágenes, que es lo que el usuario prefiere ver primero, y si las imágenes del proyecto le parecen interesantes, entonces se irá a buscar la información de éste, en el botón de la parte superior izquierda que hace que se cambie al *slider* de información:
  - o La parte correspondiente a las imágenes del lead:



- La parte de información del lead:



Por otro lado se encuentra la administración, se mostrará algunas de las páginas más relevantes de esta:

- Lista de usuarios: donde se pueden ver todos los usuarios y se puede acceder a sus datos.

LeadPost Francisco ▾

[Dashboard](#) [Comments](#) [Users](#) [Leads](#) [Blogs](#) [Categories](#) [Emails](#) [Permissions](#)

## Users list

Users list

10 records per page Search:

Online	Name	Last connection	Account	Status	Settings
	<a href="#">Francisco Palomares Barrios</a>	2014-06-22 11:20:53	SuperAdmin	Active	<a href="#">User Settings</a>
	<a href="#">Jordi Gomar Manresa</a>	2014-06-22 09:24:26	Administrator	Active	<a href="#">User Settings</a>
	<a href="#">Aitor Iturraide Martin</a>	2014-06-18 22:03:16	Administrator	Active	<a href="#">User Settings</a>
	<a href="#">Sergi Valero Muñoz</a>	2014-06-20 07:20:54	Administrator	Active	<a href="#">User Settings</a>
	<a href="#">Aida El Kabbaj</a>	2014-06-18 22:03:22	Collaborator	Active	<a href="#">User Settings</a>
	<a href="#">Maria Gelabert Paris</a>	2014-06-18 22:03:25	Collaborator	Active	<a href="#">User Settings</a>
	<a href="#">Paula Busquier</a>	2014-06-19 16:54:12	Collaborator	Active	<a href="#">User Settings</a>
	<a href="#">Javier Torre-Marin</a>	2014-06-22 01:39:38	Collaborator	Active	<a href="#">User Settings</a>
	<a href="#">Laia Garcia</a>	2014-04-14 18:42:11	Free	Active	<a href="#">User Settings</a>
	<a href="#">oscar farres</a>	2014-05-27 11:40:02	Free	Active	<a href="#">User Settings</a>

Showing 1 to 10 of 12 entries

Previous **1** 2 Next

- Opciones de usuario: donde se visualizan los datos básicos del usuario. Se puede cambiar el rol del usuario según los poderes administrativos que se tengan. Se visualiza los leads que ha visto, los que ha subido, a los que ha accedido para mayor información. Se puede ver los usuarios que ha visitado y por los que ha sido visitado. Y los detalles de todas sus conexiones.

LeadPost Francisco ▾

[Dashboard](#) [Comments](#) [Users](#) [Leads](#) [Blogs](#) [Categories](#) [Emails](#) [Permissions](#)

## Francisco Palomares Barrios •

User's avatar

User's data

Nombre	Descripcion	Género	Date of birth
Francisco Palomares Barrios	Multimedia Designer	male	1992-09-05

Address	Email	Web	Phone
Terrassa, Spain	kikopalomares@kpb.es	http://kpb.es/	

User's Rol

SuperAdmin ▾

User's leads

Views: 126 Views original post: 4 Uploaded: 12

User's leads view

10 records per page Search:

Title	date
<a href="#">Manifold House</a>	2014-06-17 16:13:40
<a href="#">Digital Culture Center</a>	2014-05-23 17:35:56
<a href="#">Gydlina Port Pilot Station</a>	2014-05-23 17:33:53
<a href="#">School of Technology and Management</a>	2014-05-23 17:31:45
<a href="#">Manifold House</a>	2014-05-23 17:28:44
<a href="#">Kengo Kuma's sublime addition</a>	2014-05-23 17:11:29
<a href="#">Digital Culture Center</a>	2014-05-22 17:10:09
<a href="#">Chaoyang city landscape</a>	2014-05-18 18:40:17

User's user

Has visited Has been visited: 40

The user has visited the profile of:

10 records per page Search:

User	date
<a href="#">oscar farres</a>	2014-05-24 15:11:56
<a href="#">Paula Busquier</a>	2014-05-22 17:11:07
<a href="#">Maria Gelabert Paris</a>	2014-05-22 17:11:06
<a href="#">Aida El Kabbaj</a>	2014-05-22 17:11:05
<a href="#">Sergi Valero Muñoz</a>	2014-05-22 17:11:04
<a href="#">Jordi Gomar Manresa</a>	2014-05-22 17:11:03
<a href="#">Aitor Iturraide Martin</a>	2014-05-22 17:11:03
<a href="#">Javier Torre-Marin</a>	2014-05-22 17:10:21

- Lista de leads: se ven todos los leads, se puede ordenar por nombre, arquitecto, puntuación, etc. Se accede a su edición, o se puede eliminar según los poderes administrativos que se tengan.

LeadPost Francisco ▾

**Blog's Leads list**

Blog's Leads list

10 records per page Search:

Visible	Name	Architect	Score	Views	+Info	Date	Edit
	<a href="#">Yamazaki Kentaro's unfinished house</a>	Yamazaki Kentaro	6.3	17	1	2014-05-04 18:31:21	<a href="#">Edit</a> <a href="#">Delete</a>
	<a href="#">Manifold House</a>	Aaron Neubert + ANX architects	9.3	28	2	2014-05-01 08:00:00	<a href="#">Edit</a> <a href="#">Delete</a>
	<a href="#">Fourth extension of Helvetia insurance headquarters</a>	Herzog & Demeuron	2.4	4	1	2014-05-01 08:00:00	<a href="#">Edit</a> <a href="#">Delete</a>
	<a href="#">In honor of Adolf Loos - Reform of a penthouse in Stockholm</a>	In praise of shadows Arkitektur	4.4	12	1	2014-05-01 08:00:00	<a href="#">Edit</a> <a href="#">Delete</a>
	<a href="#">Digital Culture Center</a>	Julio Amezcua + Francisco Pardo	3.6	14	1	2014-05-01 08:00:00	<a href="#">Edit</a> <a href="#">Delete</a>
	<a href="#">Chaoyang city landscape</a>	MAD architects	9.4	7	3	2014-04-30 19:12:00	<a href="#">Edit</a> <a href="#">Delete</a>
	<a href="#">Cave-like meeting rooms at Bond University architecture faculty</a>	CRAB Studio	4.1	3	1	2014-04-29 08:00:00	<a href="#">Edit</a> <a href="#">Delete</a>
	<a href="#">Screens house in miramar</a>	e 348 arquitectura	10.2	9	1	2014-04-28 08:00:00	<a href="#">Edit</a> <a href="#">Delete</a>
	<a href="#">Gymnasium Arena do Morro, Brazil</a>	Herzog & de Meuron	4.9	14	3	2014-04-24 00:46:34	<a href="#">Edit</a> <a href="#">Delete</a>
	<a href="#">Broadway housing</a>	Kevin daly architects	10	13	3	2014-04-23 23:19:54	<a href="#">Edit</a> <a href="#">Delete</a>

Showing 1 to 10 of 370 entries

Previous **1** 2 3 4 5 ... 37 Next

- Ranking de blogs: se puede ver el ranking de los blogs del ciclo actual, y de toda la historia.

LeadPost Francisco ▾

**Blogs Ranking**

Blogs ranking cycle

10 records per page Search:

Position	Logo	Blog	Score
1		Archdaily	795
2		Architizer	510
3		The cool hunter	240
4		Architecture Lover	45
5		The scandinavian	25
6		HIC Arquitectura	270
7		VAUMM architecture & urban planning	75
8		Archpaper	40
9		A Daily Dose of Architecture	40
10		Del tirador a la ciudad	20
11		Atasia	515
12		Dezeen	290
13		Plataforma Arquitectura	155
14		Contemporist	40
15		Architect	10

Showing 1 to 15 of 15 entries

Previous **1** Next

Blogs ranking from the beginning of time

10 records per page Search:

Position	Logo	Blog	Score
1		Archdaily	24.85
2		Designboom	22
3		Architizer	19.55
4		Atasia	14.75
5		HIC Arquitectura	14
6		Dezeen	11.4
7		Plataforma Arquitectura	8.1
8		Inhabitat	6.95
9		The cool hunter	5.9
10		DiarioDESIGN	5
11		Europaconcorsi	4.4
12		Architecture Lover	4
13		Archello	3.15
14		TECTÓNICAblog	2.6
15		I Like Architecture	2.55

Showing 1 to 15 of 60 entries

Previous **1** 2 3 4 Next

- Editar lead: es igual que la página de añadir un nuevo lead, pero con los datos precargados del lead que se quiere editar. Esta página también ha sufrido bastantes cambios, primero se probó una versión más próxima al *mockup* diseñado, pero una vez que se comenzó la subida de proyectos los colaboradores se dieron cuenta de que no era la forma más cómoda para moverse por el contenido que habían rellenado, así que se optó por cambiar el diseño a un *scroll* con todos los datos.

Home
Francisco

### Edit Lead

Lead Score: 6.3  
Lead Views: 17  
Created by: Aida El Kabbaj at 2014-05-04 18:31:21  
Last updated by: Aida El Kabbaj at 2014-05-04 18:31:21

Visible:  
☒ ON

☐ Recommended

Date of publication: 2014-05-04 18:31:21

Blog: Dezeen

Architect: Yamazaki Kentaro

Project year: 2014

Address: Masuo, Kashiwa-shi, Chiba Prefecture, Japan

Link to original post: <http://www.dezeen.com/2014/05/01/house-in-kashiwa-by-yamazaki-kentaro/>

Link to the project architect on the web: <http://ykdw.org/en/>







English Español Català Français

Title in English: Yamazaki Kentaro's unfinished house

What in English: A family house designed as an open container with barely any partitions between the rooms. Provides a flexible building to feature several multi-purpose spaces to accommodate the occupants' "future possibilities".

Where in English: The project is located in Chiba Prefecture, Japan.

How in English: The construction of the building is made by forming four boxes, each box divided into two layers around a central space, a generous double-height atrium that functions as the family's living and dining room. While the lower levels of the boxes accommodate typical rooms, the upper levels can be used for different activities.

Select image Nahoko Koide
Select image Nahoko Koide
Select image Nahoko Koide
Select image Nahoko Koide
Select image Nahoko Koide
Select image Nahoko Koide

☒ Main ☒ Architecture  
☐ Main ☐ Sustainability  
☐ Main ☐ Urban space  
☐ Main ☐ Rehabilitation  
☐ Main ☐ Innovation

Tags:

Categories Subcategories Tags

Typology  
Envelope  
Structure  
Sustainability  
Spaces  
Status  
Interior Cladding  
Materials  
Furniture

Send

Comments

About Team Privacy Policy
LeadPost architecture on the move
Powered by XFB

El resto de páginas de la plataforma van adjuntas como imágenes en anexos.



## 5 Desarrollo

### 5.1 Herramientas utilizadas

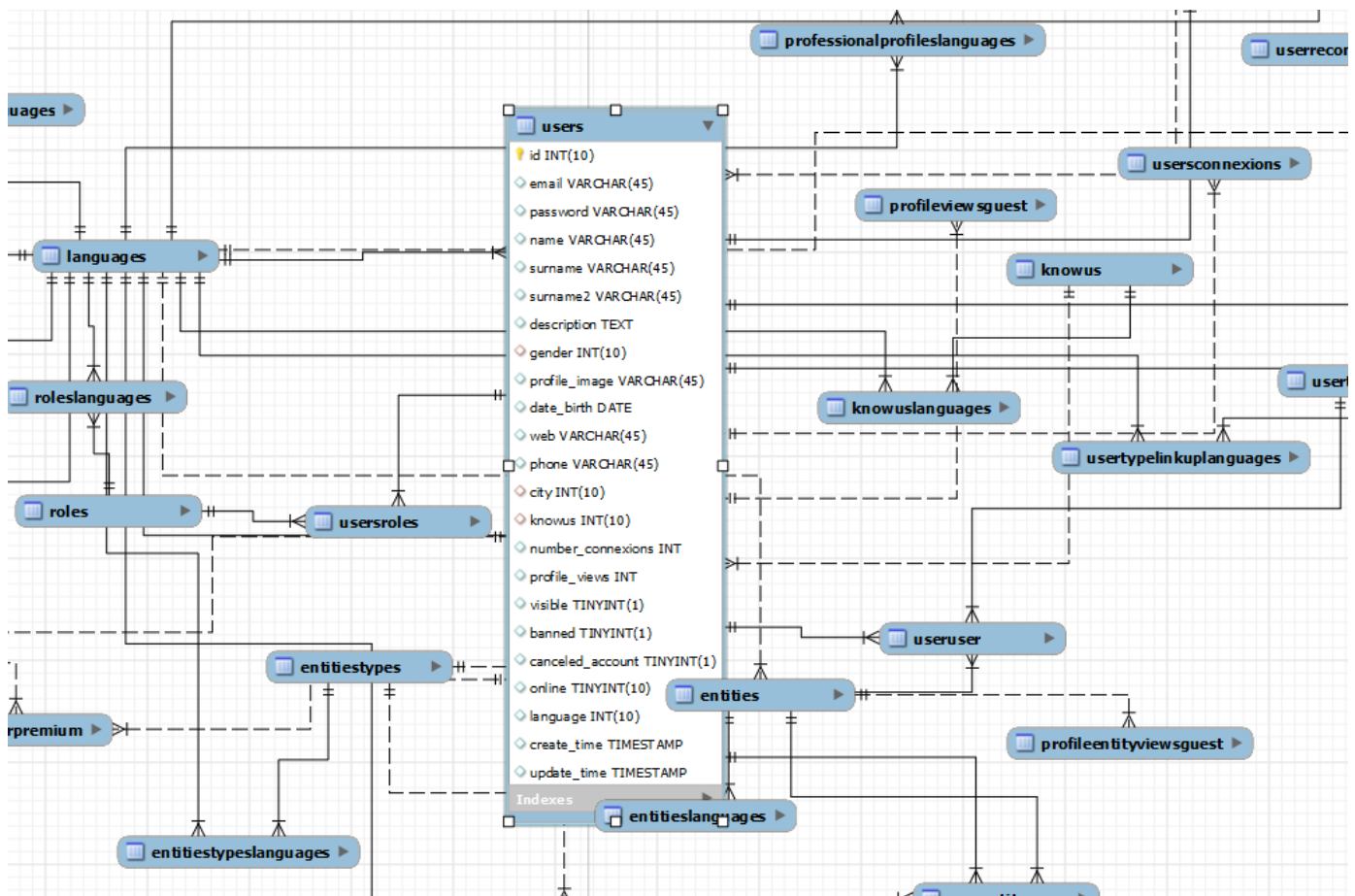
Hoy en día existe una gran variedad de herramientas y recursos para el desarrollador, tanto que elegir entre ellas no resulta una tarea sencilla. En este apartado se tratan las herramientas elegidas para desarrollar la plataforma.

#### 5.1.1 MySQL Workbench

La base de datos que precisa LeadPost, es tan grande que para realizar un simple esquema de entidad-relación sería costoso; por ese motivo se optó por usar MySQL Workbench, un híbrido entre entidad-relación y modelo relacional para llevar a cabo la gestión de la base de datos.

Aunque esta herramienta permite gestionar la base de datos del servidor desde ahí, realizando las consultas necesarias; en este caso solo se ha utilizado la parte visual que nos proporciona una manera clara de ver las relaciones de las tablas; para lo que son las consultas de la base de datos se ha usado Laravel, que se explica posteriormente en el apartado 5.3.2 Bases de datos.

Web: <http://www.mysql.com/products/workbench/>

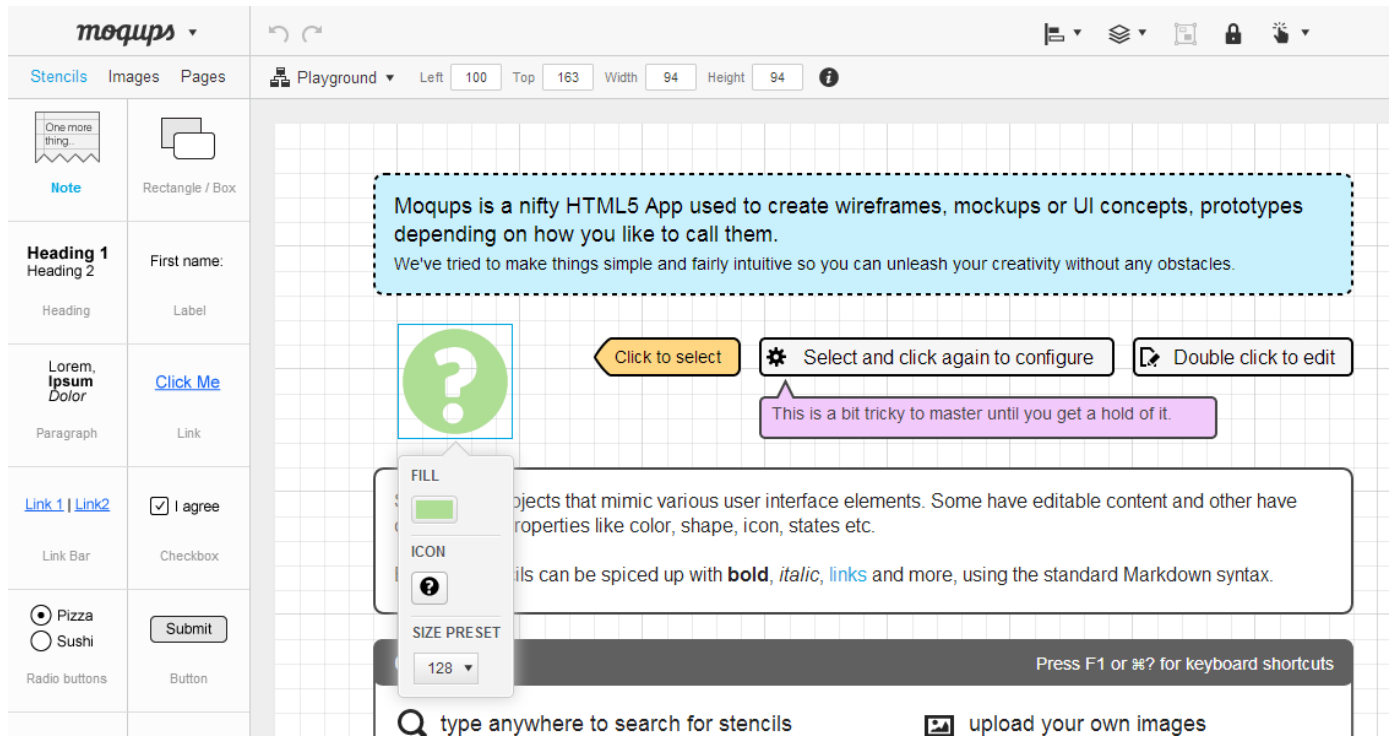


### 5.1.2 Moqups

Antes de empezar la programación, hay que tener claro el diseño que se va a emplear, y es por ello que primero se debe desarrollar los *mockups* de las diferentes páginas de la web. Existen multitud de herramientas online para crear estos *wireframes*, entre todas se escogió Moqups por las características que ofrecía su versión gratuita frente a otras.

Permite crear vectorialmente las diferentes páginas de la plataforma, además de crear una simulación de navegación por ellas.

Web: <https://moqups.com/>



### 5.1.3 Sublime Text

Sublime Text es un editor de código. Con algunas características que lo hacen muy interesante para su uso, las más destacadas, aunque tiene muchas más, son: un minimapa en que se puede ver una previsualización del código para poder desplazarse rápidamente por él; con la multi-selección se puede seleccionar diferentes partes del código al mismo tiempo; con el multi-cursor se puede crear tantos cursores como se quiera y realizar acciones todos al mismo tiempo, esto ahorra mucho tiempo.

Web: <http://www.sublimetext.com/>

```
File Edit Selection Find View Goto Tools Project Preferences Help
GROUP 1
  User.php
GROUP 2

1 <?php
2 use Illuminate\Auth\UserInterface;
3 use Illuminate\Auth\Reminders\RemindableInterface;
4
5 class User extends Eloquent implements UserInterface, RemindableInterface {
6     protected $table='users';
7     protected $fillable = array('visible', 'banned', 'canceled_account', 'email', '
      password', 'web', 'phone', 'name', 'surname', 'surname2', 'description', '
      date_birth', 'profile_image', 'language', 'number_connexions', 'address', '
      latitude', 'longitude');
8     protected $hidden = array('password');
9
10    public function getAuthIdentifier(){
11        return $this->getKey();
12    }
13
14    public function getAuthPassword(){
15        return $this->password;
16    }
17
18    public function getReminderEmail(){
19        return $this->email;
20    }
21
22    public function getRememberToken() {
23        return $this->remember_token;
24    }
25
26    public function setRememberToken($value) {
```

#### 5.1.4 Laravel

Laravel es un *framework* PHP desarrollado por Taylor Otwell. Tiene como objetivo permitir la creación de código de forma sencilla, permitiendo multitud de funcionalidades. Intenta aprovechar lo mejor de otros *frameworks* y las últimas características de PHP. Éste usa el patrón de diseño **MVC** (Modelo Vista Controlador).

##### Modelo

Laravel incluye el llamado Eloquent ORM para la creación de modelos, correspondientes a las tablas que tenga la base de datos.

##### Vista

Las vistas contienen todo el HTML de la página. Laravel contiene un sublenguaje de PHP llamado Blade, utilizado para las vistas, que convierte el código en algo más sencillo y rápido de escribir.

##### Controlador

Los controladores contienen las operaciones de la web, organizado en diferentes clases (controladores).

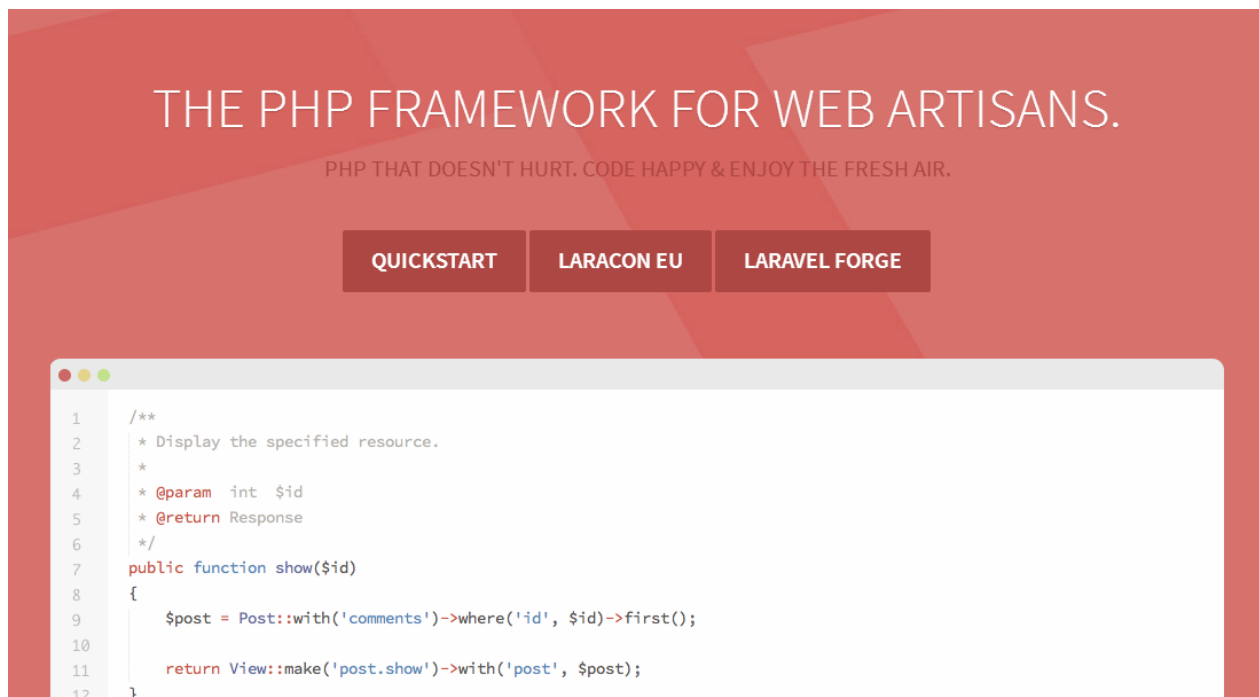
Laravel incluye un cliente de consola (**Artisan**) que nos permite ejecutar comandos del *framework*, o los que nosotros mismo creamos; para hacer por ejemplo tareas programadas en servidor.

Desde la versión 4, laravel se puede instalar directamente desde **Composer**, gestor de paquetes y dependencias de PHP. Esto nos permite modificar y agregar paquetes con el comando `composer update` desde la consola.

Laravel incorpora la posibilidad de **control de versiones de base de datos**, con un sistema de migraciones, y otro de *seeding*, para introducir datos.

Este *framework* ha crecido muy rápidamente desde su lanzamiento, y presenta una gran alternativa a la hora de programar en PHP.

Web: <http://laravel.com/>



### 5.1.5 Bootstrap

Bootstrap es un *framework* para *frontend* web. Fue creado por programadores de Twitter; y se utiliza en multitud de páginas webs, es el proyecto más popular en la web de [GitHub](https://github.com/twbs/bootstrap).

Bootstrap viene con un sistema de *grid* de doce columnas *responsive*; el cual viene preparado para la gran mayoría de tamaños de pantalla y dispositivos. Se puede configurar de modo que el máximo de ancho quede limitado por 1200px o que ocupe todo el ancho de pantalla. Estas columnas tienen cuatro variaciones de tamaño, para móviles (<768px), para tablets (≥768px), para ordenadores de escritorio (≥992px), y para ordenadores de escritorio de pantalla grande (≥1200px).

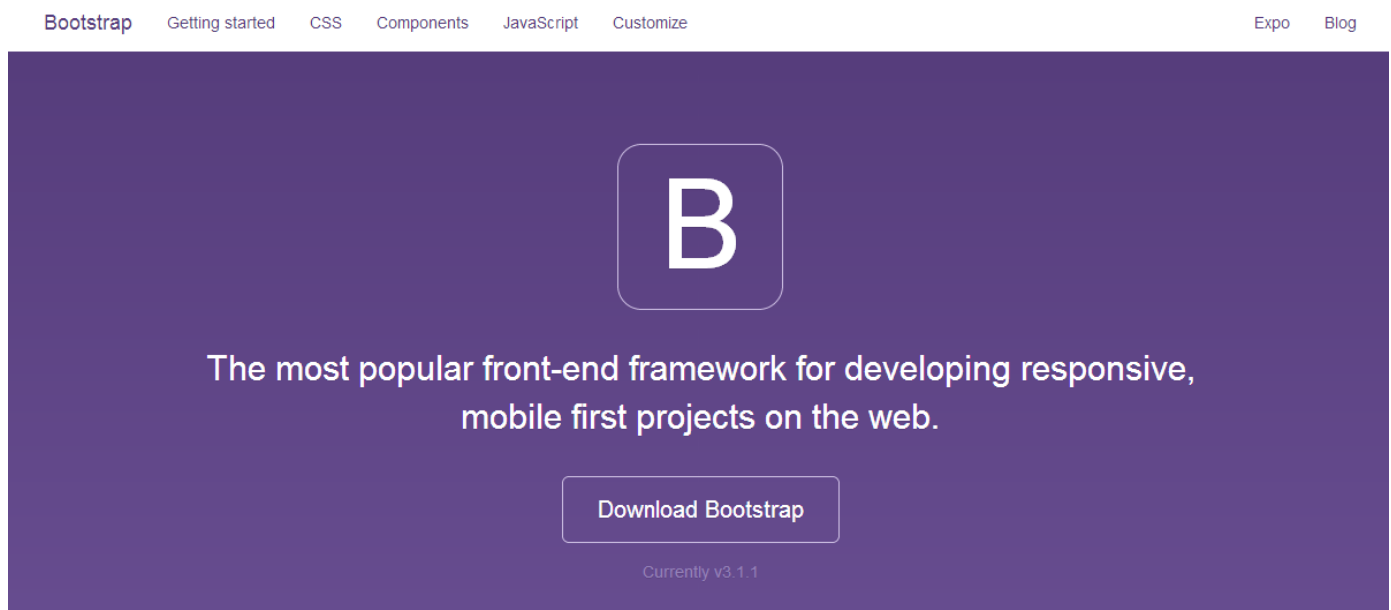
Bootstrap contiene una hoja de estilos CSS que incorpora una definición básica de estilos para todos los componentes HTML. También contiene elementos que se usan normalmente, como botones, listas desplegables, paginación, barras de progreso, etc.

Tiene componentes de Javascript basados en jQuery, que proporcionan elementos adicionales como pueden ser: sliders, menus *dropdown*, pestañas, mensajes de alertas, etc.

Bootstrap es muy personalizable, siempre se puede sobrescribir las clases que trae por defecto, o crear nuevas a partir de las que tiene.

Para su uso, se debe enlazar la hoja de estilos de bootstrap al archivo HTML; crear la estructura HTML que corresponda y asignar las clases CSS deseadas a las etiquetas HTML.

Web: <http://getbootstrap.com/>



### 5.1.6 Isotope

Para mostrar los proyectos de una forma innovadora y atractiva, se ha utilizado este plugin de jQuery. Gracias a este plugin se ha permitido ordenar las imágenes de la Home, y realizar los diferentes filtrados de búsquedas de esa forma tan atractiva.

Web: <http://isotope.metafizzy.co/>

The screenshot shows the Isotope website interface. On the left is a sidebar with navigation links: Filtering, Sorting, Layout modes..., Options, Methods, Events, License, Appendix, and FAQ. The main header features the 'Isotope' logo and the tagline 'Filter & sort magical layouts'. Below the header are three download buttons: 'Download isotope.pkgd. min.js', 'Download these docs', and 'Isotope on GitHub'. A filter bar allows selecting 'show all', 'metal', 'transition', or '-lum'. A sort bar allows selecting 'original order', 'name', 'symbol', or 'number'. A code block displays the initialization script: 

```
$container.isotope({
  itemSelector: '.element-item',
  layoutMode: 'fitRows',
  ...
})
```

. At the bottom, a grid of 18 element cards is shown, each with a symbol, atomic number, and name (e.g., Hg 80, Te 52, Bi 83, Pb 82, Au 79, K 19, Na 11, Cd 48, Ca 20, Re 75, etc.).

### 5.1.7 Otros

Los plugins a continuación listados también se han utilizado en el desarrollo de la plataforma:

- **Bxslider**: es un plugin de jQuery con el que se pueden hacer sliders de todo tipo; además están adaptados para que tengan funcionalidades en las pantallas táctiles.
- **DataTables**: es un plugin de jQuery que se encarga de gestionar el contenido de tablas; para su fácil paginación, ordenación, etc.
- **Flot**: plugin de jQuery usado para gráficos de estadísticas.
- **Font Awesome**: librería de iconos vectorizados en forma de fuente.
- **Fontello**: recopila librerías de iconos vectorizados.
- **imagesLoaded**: librería de javascript que detecta cuando las imágenes han sido cargadas.
- **jQuery**: es la librería de javascript por excelencia, utilizada en multitud de webs.
- **metisMenu**: es un plugin de jQuery y bootstrap para realizar menús.
- **Morris.js**: es un plugin de jQuery y Raphaël para realizar gráficos estadísticos.
- **Nicescroll**: es un plugin de jQuery con el que se puede personalizar las barras de *scroll*.
- **Pace**: es un plugin de jQuery que sirve para implementar barras de progreso cuando la página está haciendo una consulta a servidor.
- **Raphaël**: es una librería de javascript para dibujar gráficos en SVG.
- **TinyMCE**: es un editor de texto para incrustar en web.

## 5.2 Base de datos

El núcleo de la plataforma, se centra en su base de datos como la parte más importante. Ya que es la que permitirá las funcionalidades implementadas y por implementar de la plataforma.

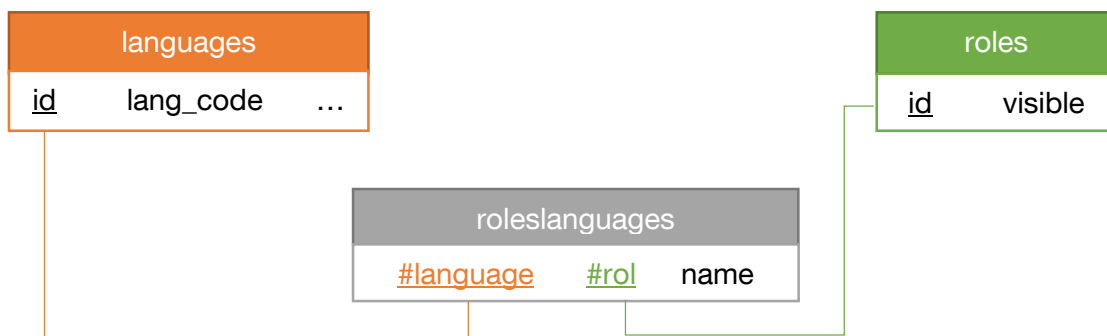
Cuando se inició el diseño de la base de datos la idea aún no estaba consolidada, de este modo no se consiguió una base sólida para su escalabilidad en un futuro. Es por ello que se realizó una primera versión, en la cual no se contemplaban muchas de las funciones que ahora necesita la plataforma.

En su primera versión, se tenían como entidades, los leads, los blogs, las categorías, los canales y los administradores. El tema de las categorías tampoco estaba muy bien organizado. Por otro lado la gestión de los diferentes idiomas no se hacía del modo más adecuado, ya que para cada idioma, por aquel entonces tres, había una columna en cada tabla donde había algo que traducir. Todos estos detalles fueron dando problemas conforme la idea de la web iba madurando e iban surgiendo nuevas necesidades.

Estas nuevas necesidades obligaron a plantear la base de datos de otro modo, empezar de nuevo. Una vez se tenía claro todas las funcionalidades que se quería en la plataforma se comenzó a diseñar una nueva.

La nueva base de datos consta de unas 113 tablas; de las cuales se explicaran las más importantes:

- Idiomas, la tabla **"languages"** es una de las más importantes, ya que prácticamente todas las demás tienen una vinculación a ésta. Esta vez las traducciones se hacen del siguiente modo: tenemos la tabla de idiomas, por otro lado la tabla de la entidad a traducir, pongamos como ejemplo la tabla de "roles", donde se recogen los roles de usuario, y por último se tiene una tabla donde se produce la vinculación entre estas dos que se suele llamar de la combinación de los nombres de ambas tablas "roleslanguages", en esta tabla se recogen las ids de las dos tablas principales además de los campos a traducir, por lo que por cada idioma habrá una fila distinta en la tabla.



- Usuarios, la tabla “**users**” recoge los datos de todos los usuarios. Los datos que se recogen en esta tabla son los siguientes:
  - **id**: el id único para identificar el usuario.
  - **email**: el email que debe ser exclusivo para el usuario; será con lo que acceda sesión a la plataforma.
  - **password**: la contraseña encriptada del usuario.
  - **name**: el nombre del usuario.
  - **surname**: el primer apellido del usuario.
  - **surname2**: el segundo apellido del usuario.
  - **description**: es la descripción que pone el usuario de sí mismo, que aparece en su perfil.
  - **gender**: vinculación a la tabla “genders” que contiene los géneros del usuario (hombre, mujer, otro).
  - **profile\_image**: es el enlace a la imagen del perfil que sube el usuario a servidor.
  - **date\_birth**: la fecha de nacimiento del usuario.
  - **web**: si el usuario tiene web propia la puede poner en su perfil.
  - **phone**: el teléfono del usuario.
  - **address**: la ciudad donde reside el usuario, se rellena con la ayuda de un autocompletar con la API de Google Maps.
  - **latitude**: gracias a la API de google Maps podemos guardar la latitud y longitud del usuario.
  - **longitude**: con la latitud y longitud del usuario guardadas podremos ofrecerle contenido personalizado según su localización.
  - **friendly\_url**: guardamos la url personalizada para el perfil de usuario, para evitar poner la id. (Ejemplo: <http://leadpost.eu/user/kikopalomares>, en vez de usar la url: <http://leadpost.eu/user/1>)
  - **knowus**: guardamos el id de la tabla “knowus” que contiene una lista de las posibles formas en que han conocido la web los usuarios registrados.
  - **number\_conexions**: el número de veces que el usuario se ha conectado.
  - **profile\_views**: las veces que ha sido visualizado el perfil de un usuario.
  - **visible**: es una booleana para ocultar un usuario y que no se pueda ver su perfil.
  - **banned**: es una booleana para indicar si el usuario ha sido baneado temporalmente (esta variable se complementa con otra tabla que recoge un registro de los motivos y el tiempo por el que ha sido baneado.)
  - **canceled\_account**: es una booleana para indicar que esa cuenta ha sido cancelada, por lo que su acceso ya no es posible.
  - **online**: booleana que indica que el usuario esta online, este campo se va actualizado con una tarea programada en servidor que va comprobando los usuarios que están conectados.
  - **language**: guarda la id del idioma que ha seleccionado el usuario para que se le muestre la interfaz.
  - **deleted\_at**: es un campo propio de Laravel para indicar que el usuario ha sido borrado de forma que no saldrá en las consultas que se realicen en la plataforma.
  - **created\_at**: es la fecha de registro del usuario.
  - **updated\_at**: este campo se va actualizando cada vez que un usuario hace una acción en la plataforma, de este modo queda registrado la última vez que estaba activo, de este campo se ayuda la tarea programada en servidor para indicar si el usuario esta online o no.



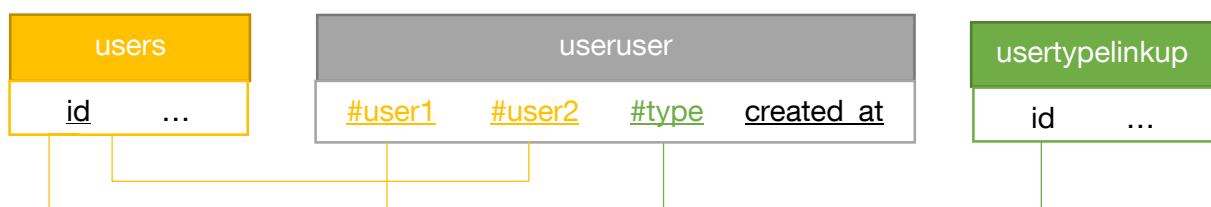
La tabla de “users” es una de las que más vinculaciones tiene, vamos a ver algunas vinculaciones de estas:

- **Conexiones de usuario:** en la tabla “usersconnexions” se van registrando todas las conexiones de los usuarios, guardando su latitud, longitud, país, IP, la plataforma (Windows, Mac, etc), el navegador, y el tiempo de entrada y salida.
- **Roles de usuario:** la tabla “usersroles” guarda los roles que tiene cada usuario; que en estos momentos puede ser: SuperAdmin, Administrador, Colaborador, Premiun y Freemium. En base a estos roles se generan una serie de filtros para bloquear partes de la plataforma, sobre todo de la administración.
- **Perfil profesional:** en la tabla “usersprofessionalprofiles” se vinculan los perfiles profesionales extraídos de la tabla “professionalprofiles” con los usuarios; en este caso el usuario puede tener tantos perfiles como desee.

Existen muchas otras tablas que guardan datos relacionado con el usuario, como por ejemplo: cuando un usuario visita un perfil de otro, o cuando un usuario mira un lead, o visita más información de éste, o cuando un visitante no registrado entra en el perfil de un usuario se guarda la IP y el usuario que ha visitado, cuando un usuario sube un lead, o lo actualiza hay un registro donde se van guardando; prácticamente todos los movimientos que va haciendo un usuario se van registrando de algún modo; para de esta forma conocer mejor al usuario y poder sugerirle proyectos, o personalizar sus búsquedas en la plataforma, u ofrecerle una publicidad que le interese.

Para permitir la escalabilidad en la base de datos, se ha empleado un modelo en muchas de las vinculaciones, consiste en tener una tabla de tipos en medio de una vinculación, permitiendo en un futuro añadir un nuevo tipo de vinculación entre dos entidades; quedará más claro con algún ejemplo:

- **Vinculación entre usuarios.** Existen muchas formas de vincular un usuario con otro: un usuario sigue a otro usuario, un usuario bloquea a otro, un usuario ha visto el perfil de otro, un usuario... etc. Para generar el menor de tablas posibles, todas las vinculaciones de un usuario con otro se produce en una sola tabla “useruser”; y se distingue el tipo de vinculación gracias a una segunda tabla que recoge los diferentes tipos de vinculación. En la tabla “useruser” se recogen las id de los dos usuarios, del primero que hace la acción sobre el segundo, se recoge el tipo de vinculación, y por último como clave primaria incluido en el grupo también se recoge la fecha, porque hay acciones que se pueden repetir, como por ejemplo la visualización del perfil de un usuario por otro que nunca se produce en el mismo instante; es por ello que la clave primaria es la combinación de estas variables.



- **Entidades:** en la tabla “entities” se recogen los datos que tienen en común los blogs, los despachos de arquitectura o las empresas; gracias a una tabla que recoge los tipos de entidades, como el modelo que se ha explicado anteriormente, en un futuro se podría añadir una nueva, como por ejemplo Universidades. Todas las entidades se controlan desde uno o más usuarios, es decir, no es un usuario en sí la entidad, es una página que controlan usuarios administradores de esta. Los atributos que se guardan son los siguientes:
  - **id:** el identificador único de la entidad.
  - **type:** el tipo de entidad que se trata (blog, despacho o empresa).
  - **entity\_name:** el nombre de dicha entidad.
  - **profile\_image:** la imagen de perfil de la entidad.
  - **web:** la página web que tenga.
  - **webviews:** la cantidad de visitas que ha tenido en su web a través de LeadPost.
  - **email:** de contacto.
  - **phone:** teléfono de contacto.
  - **language:** el idioma para la interfaz.
  - **score:** es la puntuación absoluta que tendría esta entidad en el ranking que genera la página, en estos momentos es solo para entidades tipo blog, pero en algún momento se podría aplicar a despachos o empresas.
  - **visible:** una booleana que permite poner como no visible un perfil de entidad.

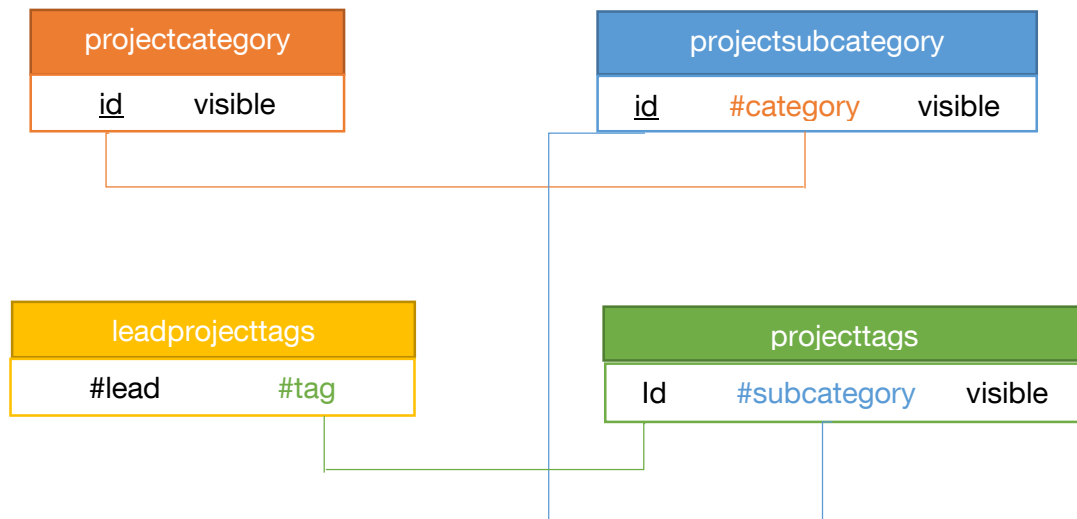
La tabla de entidades va vinculada a la de idiomas mediante otra tabla, de forma que allí pueden poner la descripción de la entidad en varios idiomas.

También existe una tabla “userentitytypelinkup” que define los tipos de vinculación entre usuarios y entidad que se produce en la tabla “userentity”, otra vez el mismo modelo explicado anteriormente; ya que en este caso hay muchas formas de vinculaciones de usuarios con entidades: usuario creador de la entidad, usuario que ha actualizado datos de ésta, usuarios que siguen a esta entidad, usuarios que administran la entidad, usuarios... etc. De este modo todas estas vinculaciones pensadas y algunas que surgirán en el futuro quedan recogidas en tan solo dos tablas.

- **Empresas:** para las entidades de tipo “empresa” existe una tabla que complementa los datos que se recogen en la tabla de “entities”, la tabla es “companies”; y consta de los siguientes atributos:
  - **#entity:** como clave foránea de la tabla de entidades.
  - **phone2:** un segundo teléfono de contacto.
  - **CIF:** de la empresa.
  - **street:** la calle donde está ubicada.
  - **number:** número de edificio.
  - **postcode:** el código postal.
  - **city.**
  - **#package:** el id del paquete comprado por la empresa.
  - **src\_logo:** el enlace a la imagen del logo subida al servidor.
- **Paquetes de compra:** existe una tabla “packages” en la que se recogen las diferentes ofertas que se ofrecen a las empresas. Una vez más tenemos la tabla que define los tipos de paquete, ya que puede haber paquetes de compra por número de impactos o por número de clicks; dependiendo de esto el paquete tendrá valores en otras dos tablas que recogen el número de impactos o de clicks según el tipo de paquete. También tiene una vinculación con los idiomas para traducir el nombre y la descripción del paquete. Sus atributos más relevantes son el precio, la periodicidad con que tienen que pagar, el número de productos que pueden subir, el número de veces que aparecen en la newsletter de LeadPost, etc.

Se explicará la parte de categorización de leads:

- **Canales:** los canales son las cinco grandes categorías que se muestran en la parte superior de la página, se recogen en la tabla “channels”, y los leads pueden ir vinculados a uno o más, indicando cuál de ellos es el canal principal. En este caso volvemos a tener tipos de canales, por si son canales del apartado Home, Social o Productos.
- **Categorías:** las categorías, tabla “projectcategory”, las subcategorías, tabla “projectsubcategory”, y las etiquetas, tabla “projecttags”, van vinculadas a los lead a través de las etiquetas tal como se puede ver en el siguiente esquema:



A continuación se explicarán las partes más relevantes referente a la publicación de leads.

- **Leads:** la tabla “leads” contiene los datos básicos que tienen en común todos los tipos de lead:
  - o **id:** el identificador único del lead.
  - o **type:** el tipo de lead que viene de otra tabla siguiendo el mismo modelo que otras veces; puede haber leads de proyectos o lead de productos.
  - o **link\_post\_product:** es el enlace para más información sobre el proyecto o producto, de la web del blog o empresa.
  - o **year:** el año de creación del proyecto o producto.
  - o **score:** la puntuación que lleva acumulada el lead por visitas, compartir en redes sociales, etc.
  - o **view:** la cantidad de veces que ha sido visto el lead.
  - o **vieworiginalpost:** la cantidad de veces que se ha accedido a más información.
  - o **recommended:** es una booleana que indica si un lead es recomendado o no por LeadPost.
  - o **visible:** es una booleana para poner como visible o no un lead.
- Se define una tabla “**userlead**” que guarda las diferentes vinculaciones de los usuarios con los leads a través de otra tabla de tipos, por ejemplo cuando un usuario sube un lead se queda guardado como autor, o cuando se edita un lead se queda como última persona que ha editado un lead, o cuando visualiza un lead, o se accede a más información, etc. Todos estos datos quedan guardados en esta tabla. De igual forma que con los usuarios se vincula los leads con las entidades, en la tabla “**entitylead**”, y vuelve haber varios tipos de vinculaciones: entidad creadora del lead, entidad como arquitecto del lead, entidad que ha visualizado el lead, etc.

- Los lead también se vincula con la tabla de idiomas para traducir sus diferentes partes, siguiendo el modelo explicado al inicio del apartado en idiomas; la tabla donde se produce la traducción se nombra “**leadprojectslanguages**”.
- Existe una tabla que relaciona los lead con las redes sociales, de forma que se contabiliza la cantidad de veces que ha sido compartido un lead en una red social determinada.
- Los lead se vinculan a uno o más canales marcando uno como canal principal. En la tabla “**leadschannels**”.
- También se vinculan a una infinidad de etiquetas, en la tabla “**leadprojecttags**”, para de este modo quedar vinculado a todos los niveles superiores de categorías.

En la base de datos se tiene en cuenta el **ranking de blogs**, primero se explicará cómo funciona exactamente y luego se mostrarán las tablas correspondientes.

El ranking consiste en agrupar los quince mejores blogs de arquitectura, durante dos semanas. El ranking consta de cuatro divisiones, tres de cinco participantes y una cuarta donde se almacena el resto de blogs. La cuarta división no tiene derecho a publicar en la web en el apartado Home, solo lo harán los 15 primeros del ranking, los pertenecientes a las tres primeras divisiones. Pese a ello los blogs de la cuarta pueden publicar en el apartado social, pero sus leads no se verán en el apartado principal de la página. Para las publicaciones existen tres tamaños de imagen diferentes, van en función de la división donde se encuentre el blog que publique el lead, de mayor tamaño en la primera división, y de menor en la tercera.

A través de las publicaciones que hagan los blogs en la plataforma, y las puntuaciones que estas obtengan, ya sea por visualización, compartir en redes sociales, votos de usuarios, etc; se genera una puntuación para el blog, una puntuación válida durante dos semanas, y que por este motivo solo se tiene en cuenta las publicaciones de esas dos semanas para la puntuación.

El ranking se renueva cada dos semanas, y dos blogs de la primera división pasan a la segunda, y dos de esta suben a primera; lo mismo pasa con la segunda y tercera división; pero el intercambio entre la tercera y la cuarta es de tres blogs en vez de dos, de este modo cada dos semanas entran al ranking tres blogs nuevos con la oportunidad de publicar en el apartado Home de la plataforma.

Ahora con una idea de cómo funciona el ranking, se pasará a explicar las tablas correspondientes a este:

- Ciclos: tabla “cycles”, en esta tabla se va registrando los ciclos que se van produciendo, cada dos semanas se hace una nueva entrada en esta tabla, indicando la fecha de finalización del ciclo. También existe una vinculación a una tabla de periodicidades, para poder crear ciclos de diferentes duraciones; aunque por ahora son de dos semanas.
- La tabla “blogscore” guarda la puntuación del blog, el ciclo donde se produjo esa puntuación, el blog correspondiente, y en la división que estaba el blog en ese ciclo.
- Divisiones: la tabla “divisions” es la lista de las cuatro divisiones, con el número de blog que suben en cada cambio de ciclo, y con la clave foránea del tamaño que tienen que tener las imágenes de los lead de los blogs de esa división.

Existen muchas otras tablas y vinculaciones en la base de datos, pero las analizadas anteriormente son las más importantes. Junto a este documento, en anexos, se adjunta un archivo de MySQL Workbench con el esquema de la base de datos; además de un archivo de Excel con el modelo relacional donde se incluyen todas las tablas con una pequeña explicación de su función.

## 5.3 Programación en Laravel

A la hora de programar la web, se eligió el framework PHP Laravel, el cual proporciona muchas facilidades a la hora de programar en PHP.

### 5.3.1 Paquetes adicionales

Algunas funcionalidades que se ha necesitado para el proyecto no las incorporaba Laravel directamente, pero existe una gran cantidad de paquetes instalables creado por desarrolladores para Laravel. En este caso se ha tenido que instalar alguno. La instalación de estos paquetes en el proyecto normalmente se suele hacer mediante Composer.

- **Laravel 4 User Agent:** gracias a la clase “Agent”, podemos manejar los datos que llegan a través del agente del navegador, como saber de qué plataforma ha entrado, que navegador es, etc.  
Web: <https://github.com/KikoPalomares/Laravel-Agent>
- **Intervention Image:** es una librería para manipular las imágenes en PHP fácilmente.  
Web: <https://github.com/Intervention/image>
- **Cron:** un paquete que proporciona una forma de realizar las tareas programadas de servidor.  
Web: <https://github.com/liebig/cron>
- **LaravelLocalization:** un paquete para gestionar los idiomas en las rutas, creando rutas traducidas según el idioma.  
Web: <https://github.com/mcamara/laravel-localization>

### 5.3.2 Base de datos

Para gestionar la base de datos se ha seguido el modelo que propone Laravel de migraciones. Gracias a Artisan, el cliente de consola con el que viene Laravel, podemos crear los archivos donde se escriben las tablas de la base de datos siguiendo el Schema Builder de Laravel.

Por ejemplo para crear la tabla de usuario en la consola de comandos debemos de poner algo así:

```
php artisan migrate:make create_users_table
```

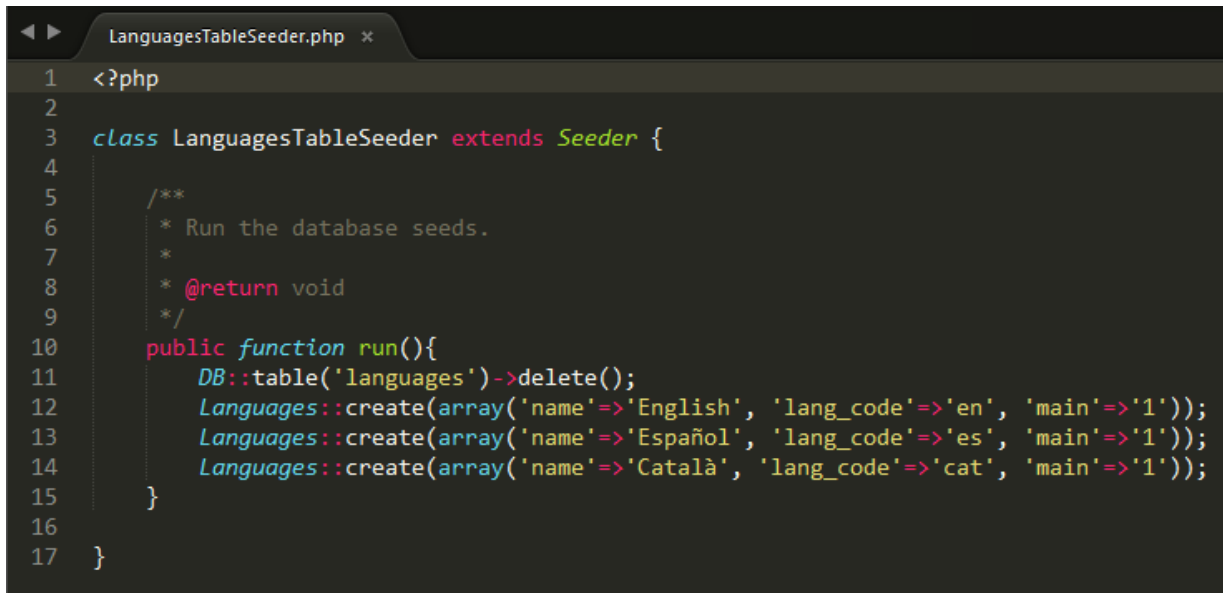
Esto nos generará un archivo donde debemos escribir el Schema de la tabla de usuarios. Se pueden añadir tantos Schemas como sea necesario, aunque es recomendable generar un archivo para cada tabla. A continuación se muestra el Schema de la tabla usuarios:

```
//tabla de usuarios
Schema::create('users', function(Blueprint $table){
    $table->engine = 'InnoDB';
    $table->increments('id');
    $table->string('email')->unique();
    $table->string('password');
    $table->string('name');
    $table->string('surname');
    $table->string('surname2');
    $table->text('description');
    $table->unsignedInteger('gender');
    $table->foreign('gender')
        ->references('id')->on('genders')
        ->onDelete('cascade')
        ->onUpdate('cascade');
    $table->string('profile_image')->default('images/profiles/default.png');
    $table->string('date_birth');
    $table->string('web');
    $table->string('phone');
    $table->string('address');
    $table->string('latitude');
    $table->string('longitude');
    $table->string('friendly_url')->unique();
    $table->unsignedInteger('know_us');
    $table->foreign('know_us')
        ->references('id')->on('knowus')
        ->onDelete('cascade')
        ->onUpdate('cascade');
    $table->integer('number_connexions')->default(0);
    $table->integer('profile_views')->default(0);
    $table->boolean('visible')->default('1');
    $table->boolean('banned')->default('0');
    $table->boolean('canceled_account')->default('0');
    $table->boolean('online')->default('0');
    $table->unsignedInteger('language');
    $table->foreign('language')
        ->references('id')->on('languages')
        ->onDelete('cascade')
        ->onUpdate('cascade')
        ->default('1');
    $table->softDeletes();
    $table->timestamps();
});
```

De este modo se van generando todas las tablas necesarias y después en la consola de comandos se introduce la siguiente orden que hace que migre la base de datos:

```
php artisan migrate
```

Laravel también proporciona un sistema de seeding de la base de datos, para introducir los datos iniciales. En la carpeta “seeds” de Laravel se colocan los archivos con los datos introducir en la base de datos. Dentro de la clase que se ejecuta realizamos la consulta de introducción de datos, como se muestra a continuación en el ejemplo de los idiomas iniciales de la plataforma:



```
1 <?php
2
3 class LanguagesTableSeeder extends Seeder {
4
5     /**
6      * Run the database seeds.
7      *
8      * @return void
9      */
10    public function run(){
11        DB::table('languages')->delete();
12        Languages::create(array('name'=>'English', 'lang_code'=>'en', 'main'=>'1'));
13        Languages::create(array('name'=>'Español', 'lang_code'=>'es', 'main'=>'1'));
14        Languages::create(array('name'=>'Català', 'lang_code'=>'cat', 'main'=>'1'));
15    }
16
17 }
```

Una vez preparadas las consultas las ejecutamos con el siguiente comando:

```
php artisan db:seed
```

Este sistema de migraciones de Laravel permite tener varias versiones de la base de datos, y tirar atrás cuando sea necesario.

### 5.3.3 Rutas

En Laravel hay que definir las rutas de la aplicación, la cual llamará a los recursos necesarios, así como a un controlador o como una vista directamente. Algunos ejemplos de rutas de LeadPost son las siguientes:

Una ruta que devuelve una vista:

```
20 //Página principal de la web
21 Route::get('/', function(){
22     return View::make('home');
23 });
24
```

Una ruta que llama a un controlador que devuelve en JSON los lead de la página principal.

```
66
67 //ruta que devuelve en JSON los datos de los leads de proyecto
68 Route::get('/showprojectlead', ['uses' => 'LeadController@showall']);
69
```

Ruta que envía por GET el id del usuario (o la friendly\_url) y lo envía al controlador que se encargará de obtener al usuario y devolver la vista.

```
52
53 //Página de perfil de usuario
54 Route::get('user/{id}', ['uses' => 'UserController@perfilUser']);
55
```

Ruta que envía por POST los datos de un formulario directamente a un controlador:

```
48
49 //añade a la bbdd los comentarios que pueden dejar en la web
50 Route::post('/newcomment', ['uses' => 'UserController@newcomment']);
51
```

### 5.3.4 Filtros

En Laravel se pueden crear filtros de modo que añadiéndolos a las rutas podemos controlar el acceso de usuarios a ciertos sitios de la web, como por ejemplo la administración.

Por ejemplo en esta ruta se define que se ha de pasar por el filtro “collaborator” para poder continuar.

```
43 Route::get('/uploadlead', ['before' => 'collaborator', function(){
44     return View::make('users.uploadLead');
45 }]);
46
```

Hay filtros que vienen por defecto en Laravel como lo es el de estar autenticado; aunque en este caso se le ha hecho alguna modificación para que devuelva un mensaje de error.

```
35
36 Route::filter('auth', function(){
37
38     $msg='You must log in first.';
39     if (Auth::guest()) return Redirect::guest('/')->with('msgdanger', $msg);
40 });
41
```



El resto de filtros creados se aplican para restringir zonas a ciertos tipos de usuarios, los filtros son los siguientes:

- **superadmin:** solo puede acceder los usuarios con este rol.

```

70
71 Route::filter('superadmin', function(){
72     if (Auth::guest()){
73         return Redirect::guest('/')->with('msgdanger', 'You must log in first.');
```

- **administrator:** pueden acceder los que tengan rol de administrado y superadministrador.

```

88 Route::filter('administrator', function(){
89
90     if (Auth::guest()){
91         return Redirect::guest('/')->with('msgdanger', 'You must log in first.');
```

- **collaborator:** acceden los usuarios con rol de colaborador, administrador y superadministrador.

```

106
107 Route::filter('collaborator', function(){
108     if (Auth::guest()){
109         return Redirect::guest('/')->with('msgdanger', 'You must log in first.');
```

- **premium:** acceden los usuarios premium, además de los colaboradores, administradores y superadministradores.

```

124
125 Route::filter('premium', function(){
126     if (Auth::guest()){
127         return Redirect::guest('/')->with('msgdanger', 'You must log in first.');
```

### 5.3.5 Modelos

Laravel utiliza el sistema de MVC (Modelo Vista Controlador), en este caso los modelos son las entidades principales de la base de datos. Estos modelos son una clase PHP donde se definen la tabla a la que pertenece, los campos de la tabla que son editables, los campos que son ocultos como las contraseñas, y se definen funciones públicas que enlazan el modelo a otros modelos, según estén vinculados en la base de datos. Por ejemplo, el modelo usuario es de la siguiente forma:

```

1  <?php
2  use Illuminate\Auth\UserInterface;
3  use Illuminate\Auth\Reminders\RemindableInterface;
4
5  class User extends Eloquent implements UserInterface, RemindableInterface {
6      protected $table='users';
7      protected $fillable = array('visible', 'banned', 'canceled_account', 'email', '
          password', 'web', 'phone', 'name', 'surname', 'surname2', 'description', '
          date_birth', 'profile_image', 'language', 'number_connexions', 'address', '
          latitude', 'longitude');
8      protected $hidden = array('password');
9
10     public function getAuthIdentifier(){
11         return $this->getKey();
12     }
13
14     public function getAuthPassword(){
15         return $this->password;
16     }
17
18     public function getReminderEmail(){
19         return $this->email;
20     }
21
22     public function getRememberToken() {
23         return $this->remember_token;
24     }
25
26     public function setRememberToken($value) {
27         $this->remember_token = $value;
28     }
29
30     public function getRememberTokenName() {
31         return 'remember_token';
32     }
33
34     public function roles(){
35         return $this->belongsToMany('Rol', 'usersroles', 'user', 'rol')->
            withTimestamps();
36     }
37
38     public function profiles(){
39         return $this->belongsToMany('ProfessionalProfiles', '
            usersprofessionalprofiles', 'user', 'profile')->withTimestamps();
40     }
41
42     public function social(){
43         return $this->belongsToMany('SocialNetwork', 'usersocialnetwork', 'user', '
            social_network')->withPivot('url')->withTimestamps();
44     }
45
46     public function leads(){
47         return $this->belongsToMany('Lead', 'userlead', 'user', 'lead')->withPivot('
            type')->withTimestamps();
48     }
49 }
50 >>

```

En el modelo llamado User que se corresponde con la tabla de la base de datos “users”, se añaden los campos que son editables, y el campo contraseña que además es oculto. Después tiene una serie de métodos get, que utiliza Laravel para el inicio de sesión del usuario y para el sistema de recordatorios de contraseñas. Y por último se pueden ver algunos métodos más que definen la relación que tiene el modelo con otros modelos; en este caso con el modelo Rol, para sacar los diferentes roles de usuario; con el modelo ProfessionalProfiles, para obtener los perfiles profesionales de usuario; con el modelo SocialNetwork con el que obtenemos las redes sociales que tiene el usuario; y por último con el modelo Lead, con el que se obtienen los leads relacionados con el usuario de un modo u otro, según el “type”.

La lista de los modelos utilizados hasta el momento es la siguiente:

- Channel
- Comment
- Currency
- Cycle
- Email
- Entity
- KnowUs
- Languages
- Lead
- LeadImages
- ProfessionalProfiles
- ProjectCategory
- ProjectSubcategory
- ProjectTag
- Rol
- SocialNetwork
- User
- UserConnexion

### 5.3.6 Controladores

Siguiendo con el modelo MVC, en Laravel se deben de crear los controladores donde se realizaran las operaciones y las consultas a la base de datos. Un controlador es una clase PHP donde se formulan métodos públicos. Por ejemplo a continuación se explican los métodos más relevantes del controlador de usuarios:

En primer lugar tenemos un método para los usuarios que escriban en los comentarios de la web:

```

1  <?php
2  class UserController extends BaseController {
3
4      public function newcomment(){
5          DB::table('comments')->insert([
6              array('user'=>Input::get('email'), 'comment'=>Input::get('comment'), '
              created_at' => date('Y-m-d H:i:s'), 'updated_at' => date('Y-m-d H:i:s'
              )),
7          ]);
8
9          $user1 = array(
10             'email'=>'kikopalomares@kpb.es',
11             'name'=>'Francisco Palomares Barrios',
12         );
13         $data = array(
14             'email' => Input::get('email'),
15             'comment'=>Input::get('comment'),
16         );
17         Mail::send('emails.newcomment', $data, function($message) use ($user1){
18             $message->from('info@leadpost.eu', 'LeadPost');
19             $message->to($user1['email'], $user1['name']);
20             $users=User::all();
21             foreach ($users as $user) {
22                 $userrol=DB::table('usersroles')->where('user', $user->id)->where('rol', 1)
                ->get();
23                 if(empty($userrol)){
24                     $userrol=DB::table('usersroles')->where('user', $user->id)->where('rol'
                     , 2)->get();
25                     if(!empty($userrol)){
26                         $message->bcc($user->email);
27                     }
28                 }else{
29                     $message->bcc($user->email);
30                 }
31             }
32             $message->subject('New comment! - LeadPost');
33         });
34
35         return Redirect::to('/')->with('msgsuccess', trans('home.index.thanks'));
36     }
37

```

En primer lugar se crea el nuevo comentario en su tabla correspondiente (líneas 5-7). Se notifica a los administradores mediante un correo electrónico que un comentario ha sido publicado, para su rápida respuesta.

La función showUsers devuelve una vista de la administración donde se muestran todos los usuarios en forma de lista.

```

98     public function showUsers($order){
99         if($order=='name'){ $asc='ASC'; } else{$asc='DESC';}
100         $users = User::orderBy($order, $asc)->paginate(15);
101         return View::make('admin.listusers', array('users' => $users));
102     }
103

```

El siguiente método es para mostrar los datos del usuario en la administración; se busca al usuario por la id que recibe por GET, y se crea la nueva vista con la variable del usuario que hemos buscado.

```

38     public function showuser($id){
39         $user=User::find($id);
40         return View::make('admin.users.edituser', array('user' => $user));
41     }
42 
```

Ahora se verá por partes el método que crea un nuevo usuario en la web:

```

104     //crear el nuevo usuario
105     public function newUser(){
106         $now = date('Y-m-d H:i:s');
107         $validator = Validator::make(
108             array(
109                 'name' => Input::get('name'),
110                 'surname' => Input::get('surname'),
111                 'email' => Input::get('email'),
112                 'address' => Input::get('address'),
113                 'latitude' => Input::get('latitude'),
114                 'longitude' => Input::get('longitude'),
115                 'password' => Input::get('password'),
116                 'profile' => Input::get('profile'),
117                 'know_us' => Input::get('know_us'),
118                 'terms' => Input::get('terms'),
119             ),
120             array(
121                 'name' => 'required',
122                 'surname' => 'required',
123                 'email' => 'required|email|unique:users',
124                 'address' => 'required',
125                 'latitude' => 'required|numeric',
126                 'longitude' => 'required|numeric',
127                 'password' => 'required',
128                 'profile' => 'required|integer',
129                 'know_us' => 'required|integer',
130                 'terms' => 'accepted',
131             )
132         );
133         if ($validator->fails()){
134             $messages = $validator->messages();
135             return Redirect::back()->with('msgdanger', '<strong>Errors:</strong> '.$messages);
136             break;
137         }
138         else{
139             //creamos nuevo usuario en el tabla de usuario

```

En este primer trozo de código se aceptan los datos que nos llegan por POST, con los validadores de Laravel. Entonces si la comprobación falla se vuelve atrás con un mensaje con los errores producidos. En el caso que se acepten los datos se procede a la creación del nuevo usuario de la siguiente forma:

```

138         else{
139             //creamos nuevo usuario en el tabla de usuario
140             $user = new User;
141             $user->email = Input::get('email');
142             $user->password = Hash::make(Input::get('password'));
143             $user->name = Input::get('name');
144             $user->surname = Input::get('surname');
145             $user->address = Input::get('address');
146             $user->latitude = Input::get('latitude');
147             $user->longitude = Input::get('longitude');
148             $user->know_us = Input::get('know_us');
149             $user->gender = 3;
150             $user->language = Session::get('language');
151             $user->save();
152             $user->friendly_url = $user->id;
153
154             //lo registramos como usuario free
155             $user->roles()->attach(Rol::find(5));
156             //le asignamos el perfil profesional
157             $user->profiles()->attach(Rol::find(Input::get('profile')));
158
159             $user->save();
160
161             $user = array(
162                 'user_id'=>$user->id,
163                 'email'=>Input::get('email'),
164                 'name'=>Input::get('name').' '.Input::get('surname')
165             );
166             $data = array(
167                 'user_id' => $user['user_id']
168             );
169
170             Mail::send('emails.welcome', $data, function($message) use ($user){
171                 $message->from('info@leadpost.eu', 'LeadPost');
172                 $message->to($user['email'], $user['name'])->subject(Email::find(1)->
                    languages()->where('language', Session::get('language'))->pluck('
                    subject'));
173             });
174
175             App::make('AuthController')->doLogin();
176
177             return Redirect::to('/');
178         }
179     }

```

Para finalizar mandamos un email de bienvenida al usuario, que automáticamente se le envía el email en el idioma en que se haya registrado. Después del email autentificamos al usuario iniciando sesión, y devolviéndolo al inicio de la plataforma.

Cuando se accede al perfil de un usuario se ejecuta este método; que aunque en una primera instancia parece que podría ser más sencillo (algo como el anterior), no lo es ya que hay que guardar una serie de datos siempre que se visualiza un perfil; se comprueba si el usuario que lo ve está registrado o no, en ese caso se guarda como que ese usuario ha visto el perfil del otro; y en el caso que no esté registrado se guardan los datos de la IP y otros en la tabla para visitas de perfiles de invitados. También se comprueba si el usuario esta visible o no.

```

56 public function perfilUser($id){
57     $today = date('Y-m-d');
58     $now = date('Y-m-d H:i:s');
59     if(is_numeric($id)){
60         $user= User::find($id);
61     }
62     else{
63         $user= User::where('friendly_url', $id)->firstOrFail();
64     }
65     if($user->visible==1){
66         //metemos la visita de perfil si esta registrado el usuario
67         if (Auth::check()){
68             //si no es el mismo usuario el resgistrado y el de perfil sumamos uno
69             //a las visitas de perfil
70             if((Auth::user()->id) != ($user->id)){
71                 if((DB::table('useruser')->where('user1', Auth::user()->id)->where(
72                     'user2', $user->id)->where('created_at', 'like', $today.'%')->
73                     count())==0){
74                     $user->profile_views++;
75                     $user->save();
76                 }
77                 //insertamos en la bbdd el registro de la visita
78                 DB::table('useruser')->insert([
79                     array('user1'=>Auth::user()->id, 'user2'=>$user->id, 'type'=>3
80                         , 'created_at' => $now, 'updated_at' => $now),
81                 ));
82             }
83             //si es un invitado el que ve el perfil
84         }else{
85             //comprobamos si en el mismo dia ya ha visto ese perfil para sumar uno
86             //a las vistas de perfil
87             if((DB::table('profileviewsguest')->where('user', $user->id)->where('ip
88                 ', Request::getClientIp()->where('created_at', 'like', $today.'%')
89                 ->count())==0){
90                 $user->profile_views++;
91                 $user->save();
92             }
93             //metemos el resgistro de la visita en la bbdd
94             DB::table('profileviewsguest')->insert([
95                 array('user'=>$user->id, 'ip'=>Request::getClientIp(), 'created_at'
96                     => $now, 'updated_at' => $now),
97             ));
98         }
99     }
100     return View::make('users.profile', array('user' => $user));
101 }
102 else{
103     return Redirect::back()->with('msgdanger', 'User Not visible.');
```



Para editar la información del usuario se ha implementado el método `editInformation`; éste responde a dos formularios diferentes, por lo que tiene un par de `if` para identificar de donde viene para hacer una consulta u otra. A continuación de los `if` tiene el validador correspondiente, si falla regresa con los errores, y si no se actualiza la información del usuario. A continuación vemos uno de los dos `if`:

```

188     public function editInformation(){
189         if(Input::get('gender')){
190             $validator = Validator::make(
191                 array(
192                     'name' => Input::get('name'),
193                     'gender' => Input::get('gender'),
194                     'date_birth' => Input::get('date_birth'),
195                     'latitude' => Input::get('latitude'),
196                     'longitude' => Input::get('longitude')
197                 ),
198                 array(
199                     'name' => 'required',
200                     'gender' => 'required|integer',
201                     'date_birth' => 'date',
202                     'latitude' => 'required|numeric',
203                     'longitude' => 'required|numeric'
204                 )
205             );
206             if ($validator->fails()){
207                 $messages = $validator->messages();
208                 return Redirect::back()->with('msgdanger', '<strong>Errors:</strong> ' .
209                     $messages);
210                 break;
211             }else{
212                 Auth::user()->name = Input::get('name');
213                 Auth::user()->surname = Input::get('surname');
214                 Auth::user()->surname2 = Input::get('surname2');
215                 Auth::user()->gender = Input::get('gender');
216                 Auth::user()->description = Input::get('description');
217                 Auth::user()->date_birth = Input::get('date_birth');
218                 Auth::user()->address = Input::get('address');
219                 Auth::user()->latitude = Input::get('latitude');
220                 Auth::user()->longitude = Input::get('longitude');
221
222                 DB::table('usersprofessionalprofiles')->where('user', Auth::user()->id)
223                     ->delete();
224                 foreach (ProfessionalProfiles::all() as $profile) {
225                     if(Input::get('profile_'.$profile->id)){
226                         DB::table('usersprofessionalprofiles')->insert([
227                             array('user'=>Auth::user()->id, 'profile'=>$profile->id, '
228                                 created_at' => date('Y-m-d H:i:s'), 'updated_at' =>
229                                     date('Y-m-d H:i:s')),
230                             ]);
231                     }
232                 }
233             }
234         }
235     }

```



Por último está la función que actualiza la imagen de perfil del usuario; que vuelve a pasar por un validador de Laravel, comprobando que se trata de un JPG o PNG; y no superando un tamaño máximo determinado. Si pasa el validador, creamos una nueva variable imagen con la clase Image, de Intervention Image; donde la ruta es la de la imagen recién subida; se realiza una serie de operaciones para que subiendo cualquier imagen, ésta quede de 500x500 píxeles. Si es una imagen horizontal quedará espacio transparente por arriba y abajo, dejando centrada la imagen, lo mismo ocurre en el caso de que la imagen sea vertical, dejando espacio por los laterales. Por último se guarda la imagen con el id del usuario seguido de su nombre en minúsculas.

```

276 public function editProfileImage(){
277
278     $validator = Validator::make(
279         array(
280             'profile_image' => Input::file('profile_image')
281         ),
282         array(
283             'profile_image' => 'mimes:jpeg,png|max:200'
284         )
285     );
286     if ($validator->fails()){
287         $messages = $validator->messages();
288         return Redirect::back()->with('msgdanger', '<strong>Errors:</strong> '.$messages);
289         break;
290     }else{
291         // resize only the width of the imag
292         $img = Image::canvas(500, 500);
293         $imginput = Image::make(Input::file('profile_image')->getRealPath())->
            resize(500, null, true);
294         //calculamos la posicion y de la imagen a insertar, esto es por si la
            imagen es más ancha que larga.
295         $y = (500-($imginput->height))/250;
296         $img->insert($imginput, 0, $y, 'center');
297         $img->crop(500, 500, 0, 0);
298         $img->encode('png', 100);
299         $img->save('images/profiles/'.Auth::user()->id.'_'.preg_replace('([A-Za-
            z0-9])', '', mb_strtolower(trim(Auth::user()->name.Auth::user()->
            surname))).'.png');
300         //guardamos en el usuario la ruta de la imagen
301         Auth::user()->profile_image = 'images/profiles/'.Auth::user()->id.'_'.preg_
            replace('([A-Za-z0-9])', '', mb_strtolower(trim(Auth::user()->name.
            Auth::user()->surname))).'.png';
302     }
303
304     Auth::user()->save();
305     return Redirect::back()->with('msgsuccess', trans('main.updatedSuccess'));
306 }
307 }
308 ?>

```

### 5.3.7 Vistas

La parte del código visible para los usuarios son las vistas. Laravel proviene de un sublenguaje de PHP llamado blade, que se usa en las vistas. Proporciona una rápida forma de imprimir código HTML usando PHP, un ejemplo: en PHP “<?php echo \$variable; ?>” mientras que en blade esto mismo sería: “{{ \$variable }}”. Los archivos de las vistas cuentan con la extensión nombre\_archivo.blade.php.

Las vistas se han organizado en una estructura de carpetas para su fácil comprensión, que va ligado al método que se ha seguido para nombrar las rutas. En la raíz de la carpeta *views* van las vistas principales de la web y las carpetas con el resto de vistas.

- La carpeta “**emails**”, es donde se guardan las vistas de los emails que manda la plataforma a sus usuarios, por ejemplo el email de bienvenida cuando se registra un usuario.
- En “**outadmin**” van los archivos de la administración. Esta carpeta está compuesta de las siguientes carpetas que siguen la estructura del menú desplegable de la administración:
  - o Blogs: se corresponde al apartado blogs de la administración.
  - o Categories: se corresponde al apartado categorías de la administración y dentro de ella se vuelve a dividir en carpetas, según sean canales, categorías de proyecto, en el futuro categorías de empresas, etc.
  - o Emails: las vistas para editar los emails están aquí.
  - o Leads: las vistas para gestionar los leads desde la administración están en este directorio.
  - o Users: en esta carpeta se encuentran las vistas de la administración relacionadas con el usuario.

Dentro de todas las carpetas que correspondan a una entidad que ha de ser listada, hay un archivo *list.blade.php*; que se encarga de ver la lista del elemento. A igual que en todas estos directorios existe el archivo *statistics.blade.php* que son estadísticas del apartado que corresponda. De esta manera se pueden utilizar nombres iguales para archivos que se encargan de hacer lo mismo, pero a elementos diferentes.

- El directorio “**password**” contiene las dos vistas necesarias para restablecer la contraseña de un usuario; una en la que se le pide el email, y otra en la que se le pide que ponga la nueva contraseña.
- En “**pieces**” se encuentran trozos de código que se añaden a otras vistas; por ejemplo, las hojas de estilo generales van todas en un mismo archivo que se vincula al resto de vistas; lo mismo pasa con zonas más grandes como el *footer* o el *header*.
- Existe una carpeta “**users**”, donde se almacenan las vistas relacionadas con el usuario, como por ejemplo el perfil de usuario, o la subida de un lead.

Conociendo la estructura de carpetas es más fácil comprender el nombre de las rutas, en las cuales se coloca el nombre de las carpetas finalizando con el nombre de la vista. Por ejemplo la vista *team.blade.php* se encuentra en la raíz de la carpeta de vistas, y la ruta en modo relativo que permite ver esta vista es la siguiente: ‘/team’. Otro ejemplo con carpetas de por medio, las opciones de usuario de la administración: correspondiente a la vista *outadmin/users/user.php*, su ruta es la siguiente: ‘admin/users/user/{id}’.

- **Home:** todas las páginas siguen una estructura de HTML parecida a la de la Home. En el head se vinculan la vista de los estilos. Para implementar estilos en Laravel el método que proporciona blade es: `{{HTML::style('ruta')}}`.

Al comienzo de la etiqueta body, incluimos el resto de “piezas” necesarias, como lo son el *header*, igual para todas las páginas; la vista de login, que incluye el popup oculto para iniciar sesión además del de registrarse; la vista de *alerts* contiene un par de divs de alerta que se activan cuando las variables de sesión de alertas se retornan desde algún controlador. Y en el caso de la página Home se incluye el menú de los canales.

```

15 <!DOCTYPE html>
16 <!--[if IE 8]> <html lang="{{Session::get('locale')}}" class="ie8"> <![endif]-->
17 <!--[if IE 9]> <html lang="{{Session::get('locale')}}" class="ie9"> <![endif]-->
18 <!--[if !IE]><!--> <html lang="{{Session::get('locale')}}"> <!--<![endif]-->
19 <head>
20     <title>{{$title}}</title>
21
22     <!-- Meta -->
23     <meta charset="utf-8">
24     <meta name="viewport" content="width=device-width, initial-scale=1.0">
25     <meta name="description" content="{{$description}}">
26     <meta name="author" content="Francisco Palomares Barrios">
27
28     <!-- Meta Shared -->
29     <meta property="og:image" content="{{URL::to($image)}}">
30     <meta property="og:locale" content='en_EN'/>
31     <meta property="og:type" content='website'/>
32     <meta property="og:title" content='{{$title}}'/>
33     <meta property="og:description" content='{{$description}}'/>
34     <meta property="og:url" content='{{$url}}'/>
35     <meta property="og:site_name" content='LeadPost'/>
36
37     <!-- CSS Global -->
38     @include('pieces.styles')
39     <link rel="shortcut icon" href="{{ URL::to('favicon.png')}}">
40     <!-- CSS Home -->
41     {{HTML::style('css/pages/home.css')}}
42
43 </head>
44 <body>
45     <!-- Header menu -->
46     @include('pieces.header')
47     <!-- Login -->
48     @include('pieces.login')
49     <!-- Alerts -->
50     @include('pieces.alerts')
51     <!-- Channels -->
52     @include('pieces.channels')
53
54     <div class="gradienttop hidden-xs"></div>
55     <div class="gradientbottom hidden-xs"></div>
56
57     <div id="tresColumnas" class="container-fluid">
58         <div class="row height100">
59             <div class="col-sm-2 columna columna1 niceScroll hidden-xs">
60
61             </div>
62             <div class="col-sm-8 columna columna2 niceScroll" onmousewheel="
columnaScrolling(2);">
63                 <div class="padding40">
64                     <div id="leads" class="isotope"></div>

```

La página Home se compone de tres columnas con *scroll* propio; siguiendo Bootstrap se le ha dado un ancho de 2 a las columnas laterales y de 8 a la interior, que es la que contiene los leads. En estos momentos las columnas laterales están vacías, porque es donde irá la búsqueda por categorías y el ranking. La columna que contiene los leads se rellena mediante JavaScript con los proyectos, en el apartado 5.4 se verá cómo se cargan los proyectos con Isotope.

```
57     <div id="tresColumnas" class="container-fluid">
58         <div class="row height100">
59             <div class="col-sm-2 columna columna1 niceScroll hidden-xs">
60
61             </div>
62             <div class="col-sm-8 columna columna2 niceScroll" onmousewheel="
columnaScrolling(2);">
63                 <div class="padding40">
64                     <div id="leads" class="isotope"></div>
65                 </div>
66             </div>
67             <div class="col-sm-2 columna columna3 niceScroll hidden-xs">
68
69             </div>
70         </div>
71     </div>
```

## 5.4 Programación en JavaScript

Los leads que se muestran en la página Home están ordenados usando el plugin de jQuery Isotope. Cuando carga la página se activa la función **showleads()**; de JavaScript.

```

25 //función de inicio que carga los leads y genera el isotope
26 function showleads(){
27     //window.history.replaceState( {} , serverUrl, '/' );
28     $.getJSON(url, {format: "json"}, function(data) {
29         if(data!=null){
30             var $container = $('#leads').imagesLoaded( function() {
31                 $container.isotope({
32                     layoutMode : 'masonry',
33                     //isOriginTop: false,
34                     itemSelector: '.lead',
35                     transitionDuration: '1.2s',
36                     masonry: {
37                         isFitWidth: true,
38                         columnWidth: 10
39                     }
40                 });
41                 $('html').animate({ scrollTop: "0" }, "slow");
42                 $('#leads').isotope( 'remove', $('#leads').isotope('getItemElements'))
43
44                 var proyectos=readshowproject(data, true);

```

Esta función obtiene un JSON de la url que se define en la variable global: `var url=serverUrl+'/showprojectlead'`. Si se reciben datos creamos el Isotope dentro del div con id 'leads', usando el `imagesLoaded` para esperar a que las imágenes se carguen y luego mostrarlas. Aún sin haber introducido las imágenes de los leads, en la línea 41 se sube el `scroll` hasta arriba por si previamente había imágenes, y volvemos a cargar esta función. Por el mismo motivo en la siguiente línea con el método `"remove"` de Isotope borramos todos los elementos que pueda existir dentro del div `#leads`. Una vez se asegura que está vacío y creado el Isotope se crea una variable llamada `proyectos` que la igualamos a una función `'readshowproject()'`, a la que le pasamos los datos de JSON, y los devuelve en un `array` con la estructura HTML preparada para Isotope; más adelante se ve en detalle esa función.

```

45
46     if(!userCheck){
47         //Unete a LeadPost
48         var elem1=document.createElement('div');
49         var string='<div class="lead leadInfo hidden-xs"><div><a
50             href="javascript:usersignup();">'+language['registerbefirt']+'</a></div></div>';
51         $(elem1).append(string);
52         //añadimos el elem1 en la posición 0.
53         proyectos.splice(0,0, elem1);
54     }
55     if(!userCollaborator){
56         var elem=document.createElement('div');
57         var string='<div class="lead leadInfo leadInfo2"><div><a href="'+serverUrl
58             +' /collaborate">'+language['joinus']+'</a></div></div>';
59         $(elem).append(string);
60         proyectos.splice(16,0, elem);
61     }

```

Tenemos dos variables globales llamadas `userCheck` y `userCollaborator`, que son booleanas que indican si el usuario está identificado y si es un usuario colaborador (o administrador) respectivamente. Estas variables las usamos para mostrar al usuario los anuncios de registrarse y de unirse como colaborador. Si son falsas estas variables se añaden los nuevos divs al *array* de proyectos; en este caso se están añadiendo en la posición 0 y en la 16.

```

69
70     $('#leads').isotope( 'insert', proyectos);
71     });
72   }
73   });
74 }

```

Por último con el método 'insert' de Isotope cargamos los proyectos en la página.

La función **readshowproject(data, widthDate);** permite leer los datos JSON de los proyectos para transformarlos en elementos HTML dentro de un *array* compatible con Isotope. Por parámetros se le pasan los datos JSON, además de una variable llamada *widthDate*, que si es verdadera devolverá el *array* con los divs de las fechas, como se ve al entrar en la página Home, pero si es falsa no devolverá las fechas, como por ejemplo cuando se realiza una búsqueda por canales.

```

160 //función que lee el JSON y retorna el array de proyectos
161 function readshowproject(data, widthDate){
162   var proyectos = [];
163   var number= data.to - data.from
164   for(var i = 0; i < number+1; i++){
165     array_lead.push(data.data[i].id);
166     var width=parseInt(data.data[i].width);
167     var height=parseInt(data.data[i].height);
168     if(width==100 && height==100){
169       width=104;
170       height=104;
171       if((data.data[i].title).length>17){
172         data.data[i].title=(data.data[i].title).substring(0,14)+"...";
173       }
174     }
175     if(width==175 && height==100){
176       width=211;
177       height=104;
178       if((data.data[i].title).length>40){
179         data.data[i].title=(data.data[i].title).substring(0,37)+"...";
180       }
181     }
182     if(width==175 && height==175){
183       width=211;
184       height=211;
185       if((data.data[i].title).length>40){
186         data.data[i].title=(data.data[i].title).substring(0,37)+"...";
187       }
188     }

```



Se crea la variable *proyectos* que se trata del *array* que se va a retornar. En la variable *data* que contiene los datos de proyectos, existe una variable que indica hasta el proyecto que se ha llegado en número, y desde el proyecto que se ha empezado (ya que hay paginación), la variable *number* guarda la resta entre estas dos para usarse en el número de veces que hay que realizar el bucle. Dentro del bucle *for* se sacan el ancho y el largo que tienen que tener las imágenes y se guardan en las variables *width* y *height*; y en función del valor de estas variables entra en un *if* u otro que les da el valor de ancho y largo, y acorta el título del proyecto para que quepa en ese espacio y no sobresalga.

```

189     var created_at = data.data[i].created_at.split(' ');
190     if(widthDate == true){
191         if(date!=created_at[0]){
192             date = created_at[0];
193             fecha_cortada = date.split('-');
194             dia = fecha_cortada[2];
195             mes = months[fecha_cortada[1]];
196             año = fecha_cortada[0];
197             var elem=document.createElement('div');
198             var string= divdate(date, dia, mes);
199             $(elem).append(string);
200             proyectos.push(elem);
201         }
202     }
203     var elem=document.createElement('div');
204     var string=divlead(date, data.data[i].id, data.data[i].img_mini, width, height
205         , data.data[i].title, data.data[i].channel);
206     $(elem).append(string);
207     proyectos.push(elem);
208 }
209 array_lead.reverse();
210 return proyectos;

```

Se guarda la fecha recibida en JSON del proyecto hasta el nivel de día en la variable *created\_at*. Se comprueba si se quieren los resultados con fecha o no, según la variable que se explicó antes (*widthDate*); en caso verdadero se comprueba que la fecha extraída del JSON sea diferente de la fecha actual, que se ha inicializado como variable global dentro de *date*.

```

157 //fecha para los proyectos
158 var date = new Date();
159 date = date.getFullYear() + "-" + (date.getMonth() +1) + "-" + date.getDate()

```

Si pasa la condición entra y le asigna el valor de *created\_at* a la variable *date*, de este modo el siguiente proyecto que tenga la misma fecha no volverá a entrar en el *if*, y no saldrá el div de la fecha repetido. Una vez dentro del *if* se divide la fecha en variables de día, mes y año; que son enviadas a la función *divdate()*; que se encarga de retornar el div con la estructura HTML de las fechas. Por último se añade al *array* de proyectos.

Cuando se sale del *if* se crea un nuevo div donde se introduce la variable *string* que contiene la estructura HTML de un div de proyecto, que es retornada de la función *divlead()*; a la cual se le parametrizan los datos del lead. Este nuevo elemento se añade al *array* de proyectos. Cuando se sale del bucle se retorna el *array* de proyectos.

Tener estas funciones que devuelven la estructura HTML permite poder usarlas en varias funciones, que modificando en un solo sitio se modifique en todos los demás; son las siguientes:

```

148 //funcion que retorna el div de la imagen del proyecto
149 function divlead(date, id, img_mini, width, height, title, channel){
150     return '<div class="lead '+date+'"><a href="javascript:lead('+id+')"><h3><span class="blanco Ccolor'+channel+'">'+title+'<span></h3><
        /a></div>';
151 }
152 //funcion que retorna del div de la imagen de la fecha
153 function divdate(date, dia, mes){
154     return '<div class="lead leadDate hidden-xs '+date+'"><a class="enlace_'+date+'"
        href="javascript:searchDate(\''+date+'\');"><span class="dia">'+dia+'</span><
        span class="mes">'+mes+'</span></a></div>';
155 }

```

Para añadir más lead en forma de *scroll* infinito, hay una función que se activa cuando se hace *scroll* (onmousewheel) en el div central, donde se encuentran los proyectos; ésta comprueba a la altura que esta el *scroll*, y un poco antes de llegar al final de este, llama a la función de añadir nuevos leads. Los nuevos leads a añadir, pueden ser leads con un previo filtro, de un canal determinado, o de una búsqueda por nombre, es por este motivo que se comprueba si existe alguna de estas búsquedas, con variables globales que se activan cuando se realiza alguna de estas búsquedas. La función que añade lead sin filtrar es *addlead()*; a la cual se la parametriza el número de página.

```

8 //Cuando el scroll llega abajo del todo cargamos la funcion addlead() con el numero
  de página correspondiente.
9 function columnaScrolling(id){
10     if ($('.columna'+id).scrollTop() >= $(".columna"+id)[0].scrollHeight - 1500){
11         //alert($(".columna"+id)[0].scrollHeight);
12         if(globalChannel!=0){
13             pagChannel++;
14             addleadChannel(pagChannel);
15         }else if(globalSearch!=''){
16             pagSearch++;
17             addleadSearch(pagSearch);
18         }else{
19             pag++;
20             addlead(pag);
21         }
22     }
23 }
24 }

```



La función `addlead()`; recibe por parámetros la nueva página de leads que tiene que cargar, de este modo añade a la URL la página y realiza la consulta. Recibe por JSON los datos y luego los introduce con el método `'insert'` de Isotope, llamando a la función que leía el JSON y retornaba el `array` preparado (`readshowproject()`).

```

75 //funcion que añade nuevos leads segun la pagina que le llega.
76 function addlead(page){
77     //disable_scroll();
78     var newurl=url+'?page='+page;
79     $.getJSON(newurl, {format: "json"}, function(data) {
80         if(data.data!=''){
81             $('#leads').isotope( 'insert', readshowproject(data, true));
82             //$('#body').animate({ scrollTop: "+=200" }, "slow");
83         }else{
84             //no more data
85         }
86         //enable_scroll();
87     });
88 }
89

```

Isotope tiene un sistema de filtrado por clases, a los diferentes elementos se les puede añadir una clase para luego filtrar por elementos con la misma clase. Cuando se generaban los elementos de proyectos, se les añadía la clase de la fecha de la publicación del proyecto, para de ese modo más tarde poder filtrarlos con la siguiente función:

```

254 //funcion que busca por fecha en el isotope
255 function searchData(date){
256     $('.column2').animate({ scrollTop: 0 }, 900);
257     $('#leads').isotope({ filter: '.'+date });
258     //$('#.enlace_'+date).removeClass('hover');
259     //$('#.enlace_'+date).addClass('grow-rotate');
260     $('#.enlace_'+date).removeAttr('href');
261     $('#.enlace_'+date).attr( 'href', 'javascript:showAll(\''+date+'\');');
262 }
263

```

Esta función se activa pulsando en las fechas mezcladas con los proyectos.

En primer lugar se sube el `scroll` del div central a la parte superior, después se filtra con el método de Isotope con la fecha que se le pasó a la función por parámetros. Por último se cambia el enlace de la fecha seleccionada por otro que contiene la función (`showall()`) para deshacer el filtro.

La función *showall()*; recibe por parámetros la fecha de la cual ha de deshacer la búsqueda, para de este modo poder devolverle la funcionalidad de búsqueda al enlace del filtrado a deshacer.

```

264 //funcion que muestra todos en el isotope
265 function showAll(date){
266     $('#leads').isotope({ filter: '*' });
267     $('#enlace_'+date).addClass('hover');
268     $('#enlace_'+date).removeClass('grow-rotate');
269     $('#enlace_'+date).removeAttr('href');
270     $('#enlace_'+date).attr( 'href', 'javascript:searchDate(\''+date+'\');');
271 }
272

```

La función que se encarga de filtrar por canales es *searchChannels()*; a la cual se le pasa por parámetros la id del canal.

En primer lugar se aplica el filtro de mostrar todos los elementos, por si se viene de un filtro de fechas u otro. Se borran las clases *active* de los links de los canales para luego más tarde añadirse tan solo al canal del que se está realizando la búsqueda. Se comprueba si el canal que se ha pulsado es el mismo que el pulsado anteriormente, si es así, se deshace el filtro llamando a la función *showleads()* que te muestra todos los leads. Si no fuese así realizamos la nueva búsqueda; haciendo la consulta con la URL de antes pero añadiendo la id del canal. Devolviendo los datos en forma de JSON; se cargan con el mismo método que en las funciones explicadas anteriormente, borrando previamente los elementos que pudiesen existir.

```

90 function searchChannels(channel){
91     $('#leads').isotope({ filter: '*' });
92     $('[class^="channellink_"]').removeClass('active');
93     pagChannel=1;
94     if(globalChannel==channel){
95         globalChannel=0;
96         pag=1;
97         showleads();
98     }else{
99         var newurl=url+'/'+channel;
100         globalChannel=channel;
101         $('#channellink_'+channel).addClass('active');
102         $.getJSON(newurl, {format: "json"}, function(data) {
103             if(data!=null){
104                 $('#leads').isotope( 'remove', $('#leads').isotope('getItemElements'))
105                 $('#leads').isotope( 'insert', readshowproject(data));
106                 $('#html').animate({ scrollTop: "0" }, "slow");
107             }
108         });
109     }
110 }

```

Ya se explicó la función que llama a las funciones correspondientes para añadir nuevos leads en forma de *scroll* infinito, ahora se muestra la función de añadir nuevos leads por filtro de canales.

```

111 //funcion que añade nuevos leads segun la pagina que le llega del canal
    correspondiente
112 function addleadChannel(page){
113     var newurl=url+'/'+globalChannel+'?page='+page;
114     $.getJSON(newurl, {format: "json"}, function(data) {
115         if(data!=null){
116             $('#leads').isotope( 'insert', readshowproject(data, false));
117             $('#html').animate({ scrollTop: "+=200" }, "slow");
118         }
119     });
120 }

```

La función *addleadChannel()*; hace la consulta a la URL pasándole la página que ha recibido por parámetros. El resultado del JSON es leído y devuelto en forma de *array* por la función *readshowproject()*; al mismo tiempo que se añade al Isotope.

Hay un buscador para introducir texto, que va realizando la búsqueda dinámicamente, al mismo tiempo que se va introduciendo el texto. Para ello se define la función *searchLeads()*; que se asegura de que se muestran todos los leads, y borra cualquier canal que pueda estar seleccionado, para proceder con la búsqueda haciendo la consulta a la que por GET envía los datos de búsqueda y esto devuelve un JSON que vuelve a ser tratado como las veces anteriores.

```

121 //función busqueda lupa
122 function searchLeads(){
123     globalSearch = document.getElementById('search').value;
124     $('#leads').isotope({ filter: '*' });
125     $('[class^="channellink_"]').removeClass('active');
126     pagSearch=1;
127     var newurl=url+'/search/'+globalSearch;
128     $.getJSON(newurl, {format: "json"}, function(data) {
129         if(data!=null){
130             $('#leads').isotope( 'remove', $('#leads').isotope('getItemElements'))
131             $('#leads').isotope( 'insert', readshowproject(data));
132             $('#html').animate({ scrollTop: "0" }, "slow");
133         }
134     });
135 }

```

La función que añade los nuevos leads una vez que se hace *scroll* y existe el filtro de búsqueda por introducción de texto funciona de forma similar a las explicadas anteriormente, se trata de la siguiente:

```

137 function addleadSearch(page){
138     var newurl=url+'/search/'+globalSearch+'?page='+page;
139     //alert(newurl);
140     $.getJSON(newurl, {format: "json"}, function(data) {
141         if(data!=null){
142             $('#leads').isotope( 'insert', readshowproject(data, false));
143             $('html').animate({ scrollTop: "+=200" }, "slow");
144         }
145     });
146 }

```

Cuando se pulsa sobre un lead, éste se abre llamando a la función `lead()`; a la cual se le pasa por parámetros la id del lead. Se hace la consulta la URL determinada, pasándole por GET el id del lead. Los datos se devuelven en JSON. Si los datos existen se cambia el título de la página y se pone el título del proyecto; además las URL actual se cambia por una seguida de `/lead/` y el id del lead, de forma que si se entra en esa URL se accede directamente al lead abierto, de esta manera es posible compartir leads en redes sociales. Se hace aparecer el fondo oscuro transparente que se coloca detrás del lead. Luego se realiza un bucle que lee todas las imágenes que trae el lead y coloca la estructura HTML de estas imágenes en una variable `"img"`. Se crea la variable `lead` donde va todo el HTML que forma el lead, incrustando las variables extraídas del JSON para rellenar los contenidos.

```

212 //funcion que abre un lead
213 function lead(id){
214     var urlLead=serverUrl+'/viewlead/'+id;
215     $.getJSON(urlLead, {format: "json"}, function(data) {
216         if(data!=null){
217             document.title = data[0].title+' | LeadPost';
218             window.history.replaceState( {}, serverUrl, '/lead/'+id );
219             $('.darkBackground').fadeIn("slow");
220             var img='';
221             for (var i = 0; i < data[1].length; i++) {
222                 img+ '<div class="slide"><div class="container-img">  <span class="photoAutor">Image by ' +
224                     data[1][i].name+' </span></div></div>';
225             };
226             var lead='<div class="fixedCenter leadContainer hide"> <button
227                 type="button" class="btn btn-default onclick="cleanpopup();">x</
228                 button> <button type="button" onclick="changePic();" class="changePic
229                 visible-xs btn btn-default"><i class="fa fa-info-circle"></i></button> <

```

Añade el lead al *body* de la página y lo hace aparecer con un *fadeIn* de jQuery. Con el plugin bxslider crea los slider de texto y de imágenes. También se bloquea el scroll hasta que se sale del lead.

```

225     $('body').append(lead);
226     $('.leadContainer').fadeIn("slow");
227     slider = $('.openlead');
228     imgslider = $('.imgslide');
229     slider= slider.bxSlider({});
230     imgslider.bxSlider({
231         mode: 'vertical',
232     });
233     disable_scroll();
234     $('.container-img').height( $('.container-img').parent().parent().parent().height());
235 }
236 });
237 }

```

Para que los sliders funcionen usando la rueda del ratón, se han añadido unas funciones para complementar el plugin bxslider:

```

238 function nextSlide(){
239     if (event.wheelDelta >= 120){
240         slider.goToPrevSlide();
241     }else if(event.wheelDelta <= -120){
242         slider.goToNextSlide();
243     }
244 }
245 function nextImg(){
246     if (event.wheelDelta >= 120){
247         imgslider.goToPrevSlide();
248     }else if(event.wheelDelta <= -120){
249         imgslider.goToNextSlide();
250     }
251 }

```

Ambas funciones se activan con un *onmousewheel* en los divs de la información y las imágenes.

Cuando se abre un lead en una pantalla pequeña, como por ejemplo desde un móvil, tan sólo se abren las imágenes a pantalla completa con un botón en la parte superior que permite cambiar al slider de la información. Este botón responde a la siguiente función:

```

275  var picture=true;
276  //cambiar de las imagenes a la informacion y viceversa en movil
277  function changepic(){
278      if(picture==true){
279          picture=false;
280          $('.leadImg').fadeOut('fast');
281          $('.leadContent').fadeIn('fast');
282          $('.fa-info-circle').addClass('fa-picture-o');
283          $('.fa-picture-o').removeClass('fa-info-circle');
284          slider.reloadSlider();
285      }else{
286          picture=true;
287          $('.leadContent').fadeOut('fast');
288          $('.leadImg').fadeIn('fast');
289          $('.fa-picture-o').addClass('fa-info-circle');
290          $('.fa-info-circle').removeClass('fa-picture-o');
291          imgslder.reloadSlider();
292      }
293  }
294  }

```

Comprueba si están las imágenes seleccionadas y si es así las pone como ocultas y muestra la información, entonces hay que recargar los sliders cada vez que se muestran.

El inicio de sesión y el registro están accesibles desde todas las páginas, se encuentran como elementos ocultos que se muestran cuando se activan las siguientes funciones:

```

1  function userlogin() {
2      cleanpopup();
3      disable_scroll();
4      $('.darkBackground').fadeIn("fast");
5      $("#userlogin").fadeIn("fast");
6  }
7
8  function usersignup(){
9      cleanpopup();
10     disable_scroll();
11     $("#usersignup").fadeIn("fast");
12     $('.darkBackground').fadeIn("fast");
13 }
14

```

Para salir de todos los *popup*, ya sea un lead abierto, el registro o *login*, o el cambio de datos de un usuario en su perfil, se llama a la misma función: *cleanpopup()*;

```
15  function cleanpopup(){
16      enable_scroll();
17
18      $(".darkBackground").fadeOut("fast");
19
20      if ($('#usersignup').length > 0) {
21          $("#usersignup").fadeOut("fast");
22      }
23      if ($('#userlogin').length > 0) {
24          $("#userlogin").fadeOut("fast");
25      }
26
27      if ($('#leadContainer').length > 0) {
28          $(".leadContainer").remove();
29          document.title = 'LeadPost - architecture on the move';
30          window.history.replaceState( {}, serverUrl, '/' );
31      }
32
33      //Pagina del perfil
34      if ($('#basic').length > 0) {
35          $('#basic').fadeOut('fast');
36      }
37      if ($('#contact').length > 0) {
38          $('#contact').fadeOut('fast');
39      }
40
41  }
```

Vuelve a activar el *scroll*, escondiendo el fondo oscuro transparente, y luego comprueba si existe alguno de los elementos que tiene que ocultar, o en el caso del lead eliminarlo. En el caso del lead se cambia el título de la página por el original y la URL actual por la del inicio.



## 5.5 Diseño centrado en el Usuario

El diseño centrado en el usuario es un enfoque de diseño dirigido por información sobre las personas que van a utilizar el producto final.

Esta nueva filosofía comenzó en la década de los cincuenta, con el diseño industrial y militar; entonces se trataba de adaptar al ser humano el diseño de los productos. Éste respondía a un proceso de investigación antropométrica y ergonómica.

En la década de los ochenta, el profesor de la Northwestern University y cofundador de Nielsen Norman Group, Norman, fue el primero que utilizó el término User Centered System Design en el conjunto de conferencias presentadas por su equipo.

El diseño centrado en el usuario es una filosofía o un enfoque, donde el usuario es el centro de todas las decisiones a la hora de diseñar un producto.

*“No sólo diseñamos productos, diseñamos experiencias de usuario, porque no es posible entender el producto desvinculado de su uso, su contexto, o de las necesidades y motivaciones del usuario final.”<sup>1</sup>*

### 5.5.1 Evaluación heurística

Una evaluación heurística es un análisis minucioso que se realiza sobre la interfaz de un sistema por un experto en la materia; con el fin de determinar si se cumplen ciertos criterios de usabilidad en el diseño o arquitectura de la información.

Es recomendable realizar esta prueba con más de un evaluador, el número óptimo está entre tres y cinco expertos. Existen muchos principios heurísticos propuestos por muchos autores, pero los más reconocidos son los **diez principios de Nielsen**, que son los siguientes:

- Visibilidad del estado del sistema
- Similitud entre el sistema y el mundo real
- Control y libertad del usuario
- Consistencia y cumplimiento de estándares
- Prevención de errores
- Preferencia al reconocimiento que a la memorización
- Flexibilidad y eficiencia de uso
- Estética y diseño minimalista
- Ayuda ante errores
- Ayuda y documentación

Para el proyecto se ha realizado una evaluación en la que el evaluador deberá investigar la página e ir contestando positiva o negativamente a las preguntas planteadas. El documento empleado se añade como anexo junto a los documentos de los evaluadores.

En este caso se ha contado con cuatro evaluadores, graduados multimedia, con un rango de edad de entre 21 y 23 años. Todos tienen una alta experiencia de usuario, y conocen las técnicas de usabilidad, por lo que los convierte en idóneos para realizar una evaluación como esta.

A continuación se presentan las preguntas formuladas con el porcentaje de respuesta por parte de los evaluadores:



## Generales

	SI	NO
<b>¿Cuáles son los objetivos del sitio web? ¿Son concretos y bien definidos? ¿Los contenidos y servicios que ofrece se corresponden con esos objetivos?</b>	100%	
<b>¿Tiene una URL correcta, clara y fácil de recordar? ¿Y las URL de sus páginas internas? ¿Son claras y permanentes?</b>	100%	
<b>¿Muestra de forma precisa y completa qué contenidos o servicios ofrece realmente el sitio web?</b> Esto está relacionado directamente con el diseño de la página de inicio, que debe ser diferente al resto de páginas y cumplir la función de 'escaparate' del sitio.	75%	25%
<b>¿La estructura general del sitio web está orientada al usuario?</b> Los sitios web deben estructurarse pensando en el usuario, sus objetivos y necesidades. No se debe calcar la estructura interna de la empresa u organización, al usuario no le interesa cómo funciona o se organiza la empresa.	75%	25%
<b>¿El look &amp; feel general se corresponde con los objetivos, características, contenidos y servicios del sitio web?</b> Por ejemplo, los colores empleados. Aunque el significado que comunica un determinado color es muy subjetivo y dependiente de la cultura y el entorno, y por lo tanto diferente para cada usuario, ciertas combinaciones de colores ofrecen una imagen más o menos formal, seria o profesional, como pueden ser los tonos de azules con el blanco, que transmiten una imagen corporativista.	100%	
<b>¿Es coherente el diseño general del sitio web?</b> Se debe mantener una coherencia y uniformidad en las estructuras y colores de todas las páginas. Esto sirve para que el usuario no se desoriente en su navegación.	100%	
<b>¿Es reconocible el diseño general del sitio web?</b> Cuánto más se parezca el sitio web al resto de sitios web, más fácil será de usar.	50%	50%
<b>¿El sitio web se actualiza periódicamente? ¿Indica cuándo se actualiza?</b> Las fechas que se muestren en la página deben corresponderse con actualizaciones, noticias, eventos...no con la fecha del sistema del usuario.	75%	25%

## Identidad e información

	SI	NO
<b>¿Se muestra claramente la identidad de la empresa-sitio a través de todas las páginas?</b>	100%	
<b>El Logotipo, ¿es significativo, identificable y suficientemente visible?</b>	50%	50%
<b>El eslogan o tagline, ¿expresa realmente qué es la empresa y qué servicios ofrece?</b>	50%	50%
<b>¿Se ofrece algún enlace con información sobre la empresa, sitio web, 'webmaster',...?</b>	100%	
<b>¿Se proporciona mecanismos para ponerse en contacto con la empresa?</b> (email, teléfono, dirección postal, fax...)	100%	
<b>¿Se proporciona información sobre la protección de datos de carácter personal de los clientes o los derechos de autor de los contenidos del sitio web?</b>	100%	
<b>En artículos, noticias, informes...¿Se muestra claramente información sobre el autor, fuentes y fechas de creación y revisión del documento?</b>	100%	

## Lenguaje y Redacción

	SI	NO
<b>¿El sitio web habla el mismo lenguaje que sus usuarios?</b> Se debe evitar usar un lenguaje corporativista. Así mismo, hay que prestarle especial atención al idioma, y ofrecer versiones del sitio en diferentes idiomas cuando sea necesario.	100%	
<b>¿Emplea un lenguaje claro y conciso?</b>	100%	
<b>¿Es amigable, familiar y cercano?</b> Es decir, lo contrario a utilizar un lenguaje constantemente imperativo, mensajes crípticos, o tratar con "desprecio" al usuario.	100%	
<b>¿1 párrafo = 1 idea?</b> Cada párrafo es un objeto informativo. Transmita ideas, mensajes...Se deben evitar párrafos vacíos o varios mensajes en un mismo párrafo.	100%	

## Rotulado

	SI	NO
<b>Los rótulos, ¿son significativos?</b> Ejemplo: evitar rótulos del tipo "haga clic aquí".	100%	
<b>¿Usa rótulos estándar?</b> Siempre que exista un "estándar" comúnmente aceptado para el caso concreto, como "Mapa del Sitio" o "Acerca de...".	100%	
<b>¿Usa un único sistema de organización, bien definido y claro?</b> No se deben mezclar sistemas de organización diferentes. Los diferentes sistemas de organización son básicamente: alfabético, geográfico, cronológico, temático, orientado a tareas, orientado al público y orientado a metáforas.	75%	25%
<b>¿Utiliza un sistema de rotulado controlado y preciso?</b> Por ejemplo, si un enlace tiene el rótulo "Quiénes somos", no puede dirigir a una página cuyo encabezamiento sea "Acerca de", o un enlace con el rótulo "Ayuda" no puede dirigir a una página encabezada con "FAQs".	100%	
<b>El título de las páginas, ¿Es correcto? ¿Ha sido planificado?</b> Relacionado con la 'findability' del sitio web.	100%	

## Estructura y Navegación

	SI	NO
<b>La estructura de organización y navegación, ¿Es la más adecuada?</b> Hay varios tipos de estructuras: jerárquicas, hipertextual, facetada,...	100%	
<b>En el caso de estructura jerárquica, ¿Mantiene un equilibrio entre Profundidad y Anchura?</b> <b>En el caso de ser puramente hipertextual, ¿Están todos los clusters de nodos comunicados?</b> Aquí se mide la distancia entre nodos.	100%	
<b>¿Los enlaces son fácilmente reconocibles como tales? ¿su caracterización indica su estado (visitados, activos,...)?</b> Los enlaces no sólo deben reconocerse como tales, sino que su caracterización debe indicar su estado (para orientar al usuario), y ser reconocidos como una unidad (enlaces que ocupan más de una línea).	75%	25%
<b>En menús de navegación, ¿Se ha controlado el número de elementos y de términos por elemento para no producir sobrecarga memorística?</b> No se deben superar los $7 \pm 2$ elementos, ni los 2 o, como mucho, 3 términos por elemento.	100%	
<b>¿Es predecible la respuesta del sistema antes de hacer clic sobre el enlace?</b> Esto está relacionado con el nivel de significación del rótulo del enlace, aunque también con: el uso de globos de texto, información contextual (indicar formato y tamaño del documento o recurso con el que vincula el enlace), la barra de estado del navegador,...	75%	25%
<b>¿Se ha controlado que no haya enlaces que no lleven a ningún sitio?</b> Enlaces que no llevan a ningún sitio: Los enlaces rotos, y los que enlazan con la misma página que se está visualizando (por ejemplo enlaces a la "home" desde la misma página de inicio)	100%	
<b>¿Existen elementos de navegación que orienten al usuario acerca de dónde está y cómo deshacer su navegación?</b> ...como breadcrumbs, enlaces a la página de inicio,...recuerde que el logo también es recomendable que enlace con la página de inicio.	50%	50%
<b>Las imágenes enlace, ¿se reconocen como clicables? ¿Incluyen un atributo 'title' describiendo la página de destino?</b> En este sentido, también hay que cuidar que no haya imágenes que parezcan enlaces y en realidad no lo sean.	50%	50%
<b>1¿Se ha evitado la redundancia de enlaces?</b>	100%	
<b>¿Se ha controlado que no haya páginas "huérfanas"?</b> Páginas huérfanas: que aun siendo enlazadas desde otras páginas, éstas no enlacen con ninguna.	100%	

## Lay-out de la página

	SI	NO
<b>¿Se aprovechan las zonas de alta jerarquía informativa de la página para contenidos de mayor relevancia?</b> (como por ejemplo la zona central)	100%	
<b>¿Se ha evitado la sobrecarga informativa?</b> Esto se consigue haciendo un uso correcto de colores, efectos tipográficos y agrupaciones para discriminar información. Al igual que en los elementos de un menú de navegación, los grupos diferentes de objetos informativos de una página, no deberán superar el número $7 \pm 2$ .	100%	
<b>¿Es una interfaz limpia, sin ruido visual?</b>	100%	
<b>¿Existen zonas en "blanco" entre los objetos informativos de la página para poder descansar la vista?</b>	100%	
<b>¿Se hace un uso correcto del espacio visual de la página?</b> Es decir, que no se desaproveche demasiado espacio con elementos de decoración, o grandes zonas en "blanco", y que no se adjudique demasiado espacio a elementos de menor importancia.	50%	50%
<b>¿Se utiliza correctamente la jerarquía visual para expresar las relaciones del tipo "parte de" entre los elementos de la página?</b> (La jerarquía visual se utiliza para orientar al usuario)	100%	
<b>¿Se ha controlado la longitud de página?</b> Se debe evitar en la medida de lo posible el scrolling. Si la página es muy extensa, se debe fraccionar.	75%	25%

## Búsqueda

	SI	NO
<b>¿Se encuentra fácilmente accesible?</b> Es decir: directamente desde la home, y a ser posible desde todas las páginas del sitio, y colocado en la zona superior de la página.	100%	
<b>¿Es fácilmente reconocible como tal?</b>	100%	
<b>¿Permite la búsqueda avanzada?</b> (siempre y cuando, por las características del sitio web, fuera de utilidad que la ofreciera)		100%
<b>¿Muestra los resultados de la búsqueda de forma comprensible para el usuario?</b>	75%	25%
<b>¿La caja de texto es lo suficientemente ancha?</b>	75%	25%
<b>¿Asiste al usuario en caso de no poder ofrecer resultados para una consultada dada?</b>		100%

## Elementos multimedia

	SI	NO
<b>¿Las fotografías están bien recortadas? ¿Son comprensibles? ¿Se ha cuidado su resolución?</b>	75%	25%
<b>¿Las metáforas visuales son reconocibles y comprensibles por cualquier usuario?</b> (prestar especial atención a usuarios de otros países y culturas)	100%	
<b>¿El uso de imágenes o animaciones proporciona algún tipo de valor añadido?</b>	100%	
<b>¿Se ha evitado el uso de animaciones cíclicas?</b>	75%	25%

## Accesibilidad

	SI	NO
¿El tamaño de fuente se ha definido de forma relativa, o por lo menos, la fuente es lo suficientemente grande como para no dificultar la legibilidad del texto?	100%	
¿El tipo de fuente, efectos tipográficos, ancho de línea y alineación empleados facilitan la lectura?	100%	
¿Existe un alto contraste entre el color de fuente y el fondo?	75%	25%
¿Incluyen las imágenes atributos 'alt' que describan su contenido?	75%	25%
¿Es compatible el sitio web con los diferentes navegadores? ¿Se visualiza correctamente con diferentes resoluciones de pantalla? Se debe prestar atención a: JScript, CSS, tablas, fuentes...	100%	
¿Puede el usuario disfrutar de todos los contenidos del sitio web sin necesidad de tener que descargar e instalar plugins adicionales?	100%	
¿Se ha controlado el peso de la página? Se deben optimizar las imágenes, controlar el tamaño del código JScript...	25%	75%
¿Se puede imprimir la página sin problemas? Leer en pantalla es molesto, por lo que muchos usuarios preferirán imprimir las páginas para leerlas. Se debe asegurar que se puede imprimir la página (no salen partes cortadas), y que el resultado es legible.		100%

## Control y retroalimentación

	SI	NO
<b>¿Tiene el usuario todo el control sobre el interfaz?</b> Se debe evitar el uso de ventanas pop-up, ventanas que se abren a pantalla completa, banners intrusivos...	100%	
<b>¿Se informa constantemente al usuario acerca de lo que está pasando?</b> Por ejemplo, si el usuario tiene que esperar hasta que se termine una operación, la página debe mostrar un mensaje indicándole lo que está ocurriendo y que debe esperar. Añadir en el mensaje el tiempo estimado que tendrá que esperar el usuario, o una barra de progreso, ayudará al usuario en este sentido.	75%	25%
<b>¿Se informa al usuario de lo que ha pasado?</b> Por ejemplo, cuando un usuario valora un artículo o responde a una encuesta, se le debe informar de que su voto ha sido procesado correctamente.	75%	25%
<b>Cuando se produce un error, ¿se informa de forma clara y no alarmista al usuario de lo ocurrido y de cómo solucionar el problema?</b> Siempre es mejor intentar evitar que se produzcan errores a tener que informar al usuario del error.	50%	50%
<b>¿Posee el usuario libertad para actuar?</b> Se debe evitar restringir la libertad del usuario: Evite el uso de animaciones que no pueden ser "saltadas", páginas en las que desaparecen los botones de navegación del browser, no impida al usuario poder usar el botón derecho de su ratón...	100%	
<b>¿Se ha controlado el tiempo de respuesta?</b> Aunque esto tiene que ver con el peso de cada página (accesibilidad) también tiene relación con el tiempo que tarda el servidor en finalizar una tarea y responder al usuario. El tiempo máximo que esperará un usuario son 10 segundos.	100%	



Una vez analizados los resultados de la evaluación podemos extraer varias conclusiones:

La estructura de la web no es parecida al resto de las páginas webs, es por eso que sale un porcentaje bajo; es normal que haya dudas, ya que se trataba de realizar un diseño un tanto diferente al resto de páginas.

Respecto al logomarca y el eslogan quizás no se entienda muy bien y haya que replantear el diseño de éste.

Hay navegaciones que el usuario no reconoce muy bien, y a la hora de deshacer algunas de ellas puede causar confusión. Ésta parte es importante que se trabaje para mejorarla.

En cuanto al *Lay-out* de la página sale un porcentaje de 50% en el tema del control del espacio visual, esto era de esperar puesto que la evaluación se ha hecho sobre un prototipo al que le faltan dos partes muy importante que son las columnas de los laterales, que ahora mismo están en vacías, por ello puede dar la impresión de que sobra “blanco”, pero una vez que se implementen las nuevas funciones esto se solucionará.

Habría que implementar una búsqueda avanzada que permita al usuario utilizar más parámetros de búsqueda. Y por otro lado en el caso de que no se produzcan resultados no se muestra ningún tipo de mensaje o sugerencia al usuario, esto habría que arreglarlo.

En accesibilidad habría que controlar el tamaño de la página, ahora mismo el tamaño de las imágenes es igual en todas las resoluciones de pantalla, aunque se hace todo lo necesario para la adaptación guardando diferentes versiones de tamaño de las imágenes, aún no está implementado.

Cuando se producen errores, se le da al usuario respuestas, pero habría que mejorar este servicio ya que no llega a ser suficientemente claro y explícito para el usuario.

## 6 Conclusiones

Sin lugar a dudas la importancia del proyecto se centra en la base de datos, es la parte en la que más tiempo y esfuerzo se ha empleado, revisándola una y otra vez para no cometer el más mínimo error. Al final se ha obtenido un gran resultado que permite, y permitirá, desarrollar la plataforma en su máximo potencial.

Con este trabajo se ha conseguido realizar un prototipo bastante avanzado de la plataforma, el cual es accesible tanto desde escritorio como desde dispositivos móviles. A pesar de todos los problemas surgidos desde un principio por no tener claro a donde se quería llegar. Con todo ello, se llega a la conclusión de la importante que es tener a priori muy claro el objetivo al que se quiere llegar antes de comenzar el desarrollo de proyectos de tanta envergadura como éste. Es importante emplear tiempo en madurar la idea lo suficiente para que el trabajo a realizar no se haga en vano. En LeadPost se tuvo que realizar un esfuerzo enorme para volver a replantear la idea y diseñar una nueva base de datos sólida y preparada para posibles cambios en el futuro, dejando de lado lo desarrollado hasta el momento.

Este trabajo me ha permitido profundizar en temas de programación desconocidos, el más relevante ha sido Laravel, pero no ha sido el único, además de introducirme en temas como puede ser el Modelo Vista Controlador que en tantos otros entornos se utilizan. Ha sido una gran experiencia para enriquecerme en cuanto a la programación web se refiere, y lo mejor, es que eso aún no ha acabado, pues queda aún mucho trabajo por delante para desarrollar la plataforma conforme a la idea.

Aún con las funciones básicas la plataforma ya podría ser operativa como medio de divulgación arquitectónica. Los próximos pasos a seguir será centrar las funcionalidades en el usuario, hacerla más social.

## 7 Bibliografía

### 7.1 Documentación de laravel

- ⇒ Taylor Otwell. *Instalación de Laravel* [en línea] URL <<http://laravel.com/docs/installation>> [Consultado: 04/12/2013]
- ⇒ Taylor Otwell. *Configuración de Laravel* [en línea] URL <<http://laravel.com/docs/configuration>> [Consultado: 04/12/2013]
- ⇒ Taylor Otwell. *Routing en Laravel* [en línea] URL <<http://laravel.com/docs/routing>> [Consultado: 04/12/2013]
- ⇒ Taylor Otwell. *Requests & Input en Laravel* [en línea] URL <<http://laravel.com/docs/requests>> [Consultado: 15/12/2013]
- ⇒ Taylor Otwell. *Views & Responses en Laravel* [en línea] URL <<http://laravel.com/docs/responses>> [Consultado: 04/12/2013]
- ⇒ Taylor Otwell. *Controllers en Laravel* [en línea] URL <<http://laravel.com/docs/controllers>> [Consultado: 04/12/2013]
- ⇒ Taylor Otwell. *Security en Laravel* [en línea] URL <<http://laravel.com/docs/security>> [Consultado: 17/12/2013]
- ⇒ Taylor Otwell. *Forms & HTML en Laravel* [en línea] URL <<http://laravel.com/docs/html>> [Consultado: 15/12/2013]
- ⇒ Taylor Otwell. *Localization en Laravel* [en línea] URL <<http://laravel.com/docs/localization>> [Consultado: 21/12/2013]
- ⇒ Taylor Otwell. *Mail en Laravel* [en línea] URL <<http://laravel.com/docs/mail>> [Consultado: 13/01/2014]
- ⇒ Taylor Otwell. *Pagination en Laravel* [en línea] URL <<http://laravel.com/docs/pagination>> [Consultado: 07/02/2014]
- ⇒ Taylor Otwell. *Session en Laravel* [en línea] URL <<http://laravel.com/docs/session>> [Consultado: 25/02/2014]
- ⇒ Taylor Otwell. *Templates en Laravel* [en línea] URL <<http://laravel.com/docs/templates>> [Consultado: 05/12/2013]
- ⇒ Taylor Otwell. *Validation en Laravel* [en línea] URL <<http://laravel.com/docs/validation>> [Consultado: 18/03/2014]
- ⇒ Taylor Otwell. *Database en Laravel* [en línea] URL <<http://laravel.com/docs/database>> [Consultado: 05/12/2013]
- ⇒ Taylor Otwell. *Query Builder en Laravel* [en línea] URL <<http://laravel.com/docs/queries>> [Consultado: 05/12/2013]
- ⇒ Taylor Otwell. *Eloquent ORM en Laravel* [en línea] URL <<http://laravel.com/docs/eloquent>> [Consultado: 05/12/2013]
- ⇒ Taylor Otwell. *Schema Builder en Laravel* [en línea] URL <<http://laravel.com/docs/schema>> [Consultado: 20/12/2013]
- ⇒ Taylor Otwell. *Migrations & Seeding en Laravel* [en línea] URL <<http://laravel.com/docs/migrations>> [Consultado: 20/12/2013]
- ⇒ Taylor Otwell. *Commands en Laravel* [en línea] URL <<http://laravel.com/docs/commands>> [Consultado: 24/03/2014]
- ⇒ Oliver Vogel. *Installation Intervention Image* [en línea] URL <[http://intervention.olivervogel.net/getting\\_started/installation](http://intervention.olivervogel.net/getting_started/installation)> [Consultado: 16/03/2014]
- ⇒ Oliver Vogel. *Canvas Intervention Image* [en línea] URL <<http://intervention.olivervogel.net/api/canvas>> [Consultado: 16/03/2014]
- ⇒ Oliver Vogel. *Crop Intervention Image* [en línea] URL <<http://intervention.olivervogel.net/api/crop>> [Consultado: 16/03/2014]

- ⇒ Oliver Vogel. *Destroy Intervention Image* [en línea] URL <<http://intervention.olivervogel.net/api/destroy>> [Consultado: 16/03/2014]
- ⇒ Oliver Vogel. *Encode Intervention Image* [en línea] URL <<http://intervention.olivervogel.net/api/encode>> [Consultado: 16/03/2014]
- ⇒ Oliver Vogel. *Height Intervention Image* [en línea] URL <<http://intervention.olivervogel.net/api/height>> [Consultado: 16/03/2014]
- ⇒ Oliver Vogel. *Make Intervention Image* [en línea] URL <<http://intervention.olivervogel.net/api/make>> [Consultado: 16/03/2014]
- ⇒ Oliver Vogel. *Resize Intervention Image* [en línea] URL <<http://intervention.olivervogel.net/api/resize>> [Consultado: 16/03/2014]
- ⇒ Oliver Vogel. *Save Intervention Image* [en línea] URL <<http://intervention.olivervogel.net/api/save>> [Consultado: 16/03/2014]
- ⇒ Oliver Vogel. *Width Intervention Image* [en línea] URL <<http://intervention.olivervogel.net/api/width>> [Consultado: 16/03/2014]
- ⇒ Chris Sevilleja. *Simple Laravel Layouts using Blade* [en línea] URL <<http://scotch.io/tutorials/simple-laravel-layouts-using-blade>> [Consultado: 08/12/2013]
- ⇒ Jens Segers. *Laravel 4 User Agent* [en línea] URL <<https://github.com/jenssegers/Laravel-Agent>> [Consultado: 19/12/2013]
- ⇒ liebigh. *Cron* [en línea] URL <<https://github.com/liebigh/cron>> [Consultado: 02/03/2014]
- ⇒ *Laravel Cheat Sheet* [en línea] URL <<http://cheats.jesse-obrien.ca/>> [Consultado: 02/03/2014]
- ⇒ *Basic Auth in 4 Minutes* [en línea] URL <<https://laracasts.com/lessons/basic-authentication-in-four-minutes>> [Consultado: 27/12/2013]

## 7.2 Documentación de Bootstrap

- ⇒ Twitter. *Getting started* [en línea] URL <<http://getbootstrap.com/getting-started/>> [Consultado: 13/05/2013]
- ⇒ Twitter. *CSS* [en línea] URL <<http://getbootstrap.com/css/>> [Consultado: 13/05/2013]
- ⇒ Twitter. *Components* [en línea] URL <<http://getbootstrap.com/components/>> [Consultado: 13/05/2013]
- ⇒ Twitter. *JavaScript* [en línea] URL <<http://getbootstrap.com/javascript/>> [Consultado: 13/05/2013]
- ⇒ Eonasdan. *Bootstrap datetimepicker* [en línea] URL <<http://eonasdan.github.io/bootstrap-datetimepicker/>> [Consultado: 12/05/2014]
- ⇒ caseyhol. *bootstrap-select* [en línea] URL <<http://silviomoreto.github.io/bootstrap-select/3/>> [Consultado: 12/05/2014]

## 7.3 Documentación de Isotope:

- ⇒ David DeSandro. *Filtering* [en línea] URL <<http://isotope.metafizzy.co/filtering.html>> [Consultado: 17/05/2013]
- ⇒ David DeSandro. *Sorting* [en línea] URL <<http://isotope.metafizzy.co/sorting.html>> [Consultado: 17/05/2013]
- ⇒ David DeSandro. *Layout modes* [en línea] URL <<http://isotope.metafizzy.co/layout-modes.html>> [Consultado: 17/05/2013]
- ⇒ David DeSandro. *Options* [en línea] URL <<http://isotope.metafizzy.co/options.html>> [Consultado: 17/05/2013]
- ⇒ David DeSandro. *Methods* [en línea] URL <<http://isotope.metafizzy.co/methods.html>> [Consultado: 17/05/2013]
- ⇒ David DeSandro. *Events* [en línea] URL <<http://isotope.metafizzy.co/events.html>> [Consultado: 17/05/2013]

- ⇒ David DeSandro. *Appedix* [en línea] URL <<http://isotope.metafizzy.co/appendix.html>> [Consultado: 17/05/2013]

## 7.4 Evaluación heurística

- ⇒ Jakob Nielsen. *10 Usability Heuristics for User Interface Design* [en línea] URL <<http://www.nngroup.com/articles/ten-usability-heuristics/>> [Consultado: 09/05/2014]
- ⇒ Jakob Nielsen. *How to Conduct a Heuristic Evaluation* [en línea] URL <<http://www.nngroup.com/articles/how-to-conduct-a-heuristic-evaluation/>> [Consultado: 09/05/2014]
- ⇒ Martín Fernández, Francisco J. *Guía de Evaluación Heurística de Sitios Web* [en línea] URL <<http://www.nosolousabilidad.com/articulos/heuristica.htm>> [Consultado: 09/05/2014]
- ⇒ Joaquín Márquez Correa. *Guía para evaluación experta* [en línea] URL <[http://www.jmarquez.com/documentos/jm\\_checklist.pdf](http://www.jmarquez.com/documentos/jm_checklist.pdf)> [Consultado: 09/05/2014]
- ⇒ <sup>1</sup> Yusef Hassan Montero y Sergio Ortega Santamaría. *Diseño Centrado en el Usuario* [en línea] URL <<http://www.nosolousabilidad.com/manual/3.htm>> [Consultado: 09/05/2014]

## 7.5 Otras consultas

- ⇒ Dave Gandy. *Font Awesome* [en línea] URL <<http://fortawesome.github.io/Font-Awesome/icons/>> [Consultado: 18/02/2014]
- ⇒ bxCreative. *Options* [en línea] URL <<http://bxslider.com/options>> [Consultado: 23/05/2013]
- ⇒ Inuyaksa. *Nicescroll* [en línea] URL <<http://areaaperta.com/nicescroll/>> [Consultado: 29/04/2014]
- ⇒ HubSpot. *Pace.js* [en línea] URL <<http://github.hubspot.com/pace/>> [Consultado: 11/03/2014]